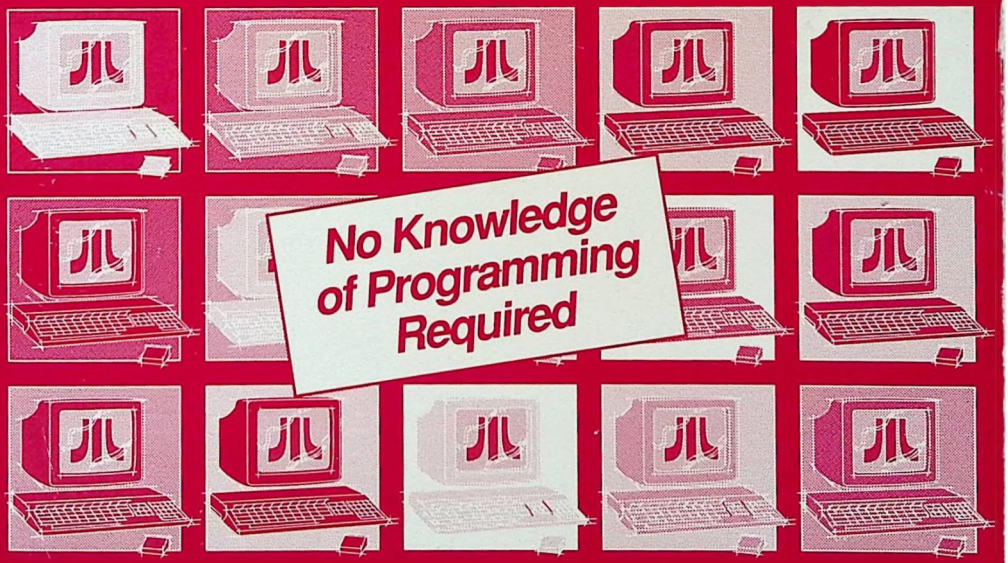
Intermediate and Advanced Atari ST Subjects



by Ralph C. Turner

Intermediate and Advanced Atari ST Subjects

Ry Ralph C. Turner

Index Legalis Publishing Company
Post Office Box 1822-3
Fairfield, IA 52556
United States of America
(515) 472-2293

Intermediate and Advanced Atari ST Subjects

Copyright © 1989 by Ralph C. Turner

First Printing 1989
Printed and bound in the United States of America.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any informational storage or retrieval system - except by a reviewer who may quote brief passages in a review to be printed in a magazine or newspaper - without permission in writing from the publisher.

ATARI, MEGA ST, ST, 1040ST, 520ST and TOS are trademarks or registered trademarks of Atari Corporation. GEM is a trademark of Digital Research Inc. IBM is a trademark of International Business Machines, Inc. MACINTOSH and MAC are trademarks or registered trademarks of Apple Computer, Inc. MS-DOS is a trademark of Microsoft Corporation. dBASE is a trademark of Ashton-Tate.

This book is sold as is, without warranty of any kind, either express or implied, respecting the contents of this book, including but not limited to implied warranties for the book's quality, performance, merchantability, or fitness for any particular purpose. Neither Index Legalis Publishing Company nor the author shall be liable to the purchaser or any other person or entity with respect to any liability, loss, or damage caused or alleged to be caused directly or indirectly by this book.

Published by:

Index Legalis Publishing Company
Post Office Box 1822-3
Fairfield, IA 52556
United States of America
(515) 472-2293

Library of Congress Cataloging-in-Publication Data

Turner, Ralph C., 1943-
Intermediate and advanced Atari ST subjects / by Ralph C. Turner.
p. cm.
Includes index.
ISBN 0-945959-01-X : $16.95
1. Atari ST (Computer) I. Title.
QA76.8.A824T88 1989
004.165--dc20 89-29731
CIP

Table of Contents

	Introduction	4
1.	Binary Code De-Mystified	7
	Acknowledgments	14
2.	ASCII, Hexadecimal and Decimal	15
3.	File Editors and Control Codes	22
4.	ASCII Files	41
5.	Disks and File Recovery	51
6.	Hard Drives	72
7.	Using Multiple Hard Drives and Assembling a Hard Drive System	104
8.	Connecting 5 1/4-inch Floppy Drives	120
9.	IBM Emulation	129
10.	Miscellaneous Subjects	134
Appendix A: ASCII Control Characters (0-31)		150
Appendix B: ASCII-decimal-hexadecimal (32-127)		151
Appendix C: ASCII-decimal-hexadecimal (128-255)		152
Appendix D: Sources for Public Domain Programs		153
Appendix E: Addresses of Suppliers		154
Index		155

INTRODUCTION

I'm not kidding. I actually received the following letter soon after my first ST book was published:

> Dear Mr. Turner:
>
> The experience of owning an ST has been an exercise in feeling dumb, and your *Atari ST Book of Tips, Instructions, Secrets and Hints* is at this early stage something that I suspect was written by Albert Einstein, for whom you are his contemporary extension. I will love your book, but only after I learn my ABCs.

With a "compliment" like that, why do I now come out with a volume dealing with *intermediate* and *advanced* topics? There are two reasons. First, in spite of that letter, many of my readers are apparently looking for more advanced information. I come to this conclusion because of the results of a survey I conducted. I asked a large sampling of the readers of *The Atari ST Book* what subjects they wanted covered in future books, and a majority of the responses requested more sophisticated topics. The second reason for including more advanced material in this book has to do with my own growth. As my interests and understanding mature, it's only natural that the subject of my writing should similarly evolve.

And anyway, "beginning," "intermediate," and "advanced" are subjective terms, at least to some degree. One user's advanced topic may be another user's rudimentary trivia. (Since you'll be the ultimate judge, don't hesitate to drop me a note saying whether this book is too simple, too complicated, or just about right. Your feedback will be useful to me in writing my next book.)

WHAT THIS BOOK IS NOT

Intermediate and Advanced Atari ST Subjects is not a rehash of the owner's manual for your ST, nor does it deal with subjects that are included in my *Atari ST Book.* In fact, so as not to waste space by including previously covered material, I've assumed that the reader of the present volume is at least somewhat familiar with most of the subjects dealt with in *The Atari ST Book.*

However, let me reassure those of you who are new ST users. Even when dealing with advanced topics, I've tried to write in such a way that the novice doesn't become lost. For example, I've totally avoided discussions of programming. *Intermediate and Advanced Atari ST Subjects* is for *all* users of the ST computers, regardless of whether they have any interest in programming.

This book is also not a comprehensive guide to commercial programs available for the ST. In fact, whenever possible, I have tried to mention *public domain* and *shareware*, rather than commercial programs. This gives you an opportunity to experiment with some of the subjects in the book without having to invest a great deal of money. Then, after you've gained some familiarity with a particular class of programs (such as file editors) you can consult the ST-specific magazines (or talk with an Atari dealer, or attend an Atari ST users' group meeting) for up-to-date information about the commercial counterparts of these programs.

CONVENTIONS USED IN *ATARI ST SUBJECTS*

Throughout this book, square brackets [ ] are used to represent keys on your ST's keyboard - for instance, [Delete], [Alternate], [Esc], etc. Function keys are not bracketed: F1, F7, etc. The four cursor keys are referred to as [Left], [Right], [Up] and [Down], while the button on the back of the ST that performs a warm boot is symbolized [Reset].

Sometimes you'll be instructed to press two keys at the same time. This type of key combination is designated with the plus character. For instance, [Control]+[X] is an instruction to press the [Control] key and hold it down, then at the same time press the [X] key.

If two keys are to be pressed in sequence, a comma "," will separate the keys. For instance, [Alternate],[Home] is an instruction to press the [Alternate] key, release it, then press the [Home] key.

"Double-clicking" refers to the process of clicking the left mouse button twice in order to select an item, whereas "mouse-clicking" refers to clicking the left mouse button only once.

ADDRESSES AND SOURCES

Sources for products mentioned in *Atair ST Subjects* are listed in three different places.

Appendix D contains a list of all the public domain and shareware programs discussed in the book. The list includes the name and location of the file on CompuServe; the name and number of the file on GEnie; and the number of the BRE Software disk that contains the file.

In some cases, where a product or commercial program is mentioned only once or twice in the book, its distributor's name and address are listed in the text where the product is discussed. For products that are mentioned numerous times throughout the book, refer to Appendix E for information about sources.

WHAT'S NEXT

I've spent over a year researching and writing *Atari ST Subjects*, and it has been an exciting project. For every question that I find an answer to, it seems that two new questions come to mind. There's only one solution: write another book. I'm already working on it.

Ralph C. Turner
c/o Index Legalis Publishing Company
Post Office Box 1822-3
Fairfield, IA 52556, U.S.A.

Chapter One

BINARY CODE DE-MYSTIFIED

QUESTION: It's my understanding that the inside of a computer consists primarily of thousands of electrical circuits. But I don't see how circuits, no matter *how* many of them there are, can understand the different letters I type on the keyboard.

ANSWER: The best way to explain this is for us to forget about computers for awhile and consider a hypothetical situation. Let's assume that you and I would like to go to the movies tonight, but you're not sure whether you'll be able to get off work in time for the 7 o'clock show. Consequently, in order for me to know whether we're going or not, you'll have to give me a "yes" or a "no" answer, sometime before this evening.

Let's further assume that, since we're both puzzle and riddle lovers, we decide to make a game out of getting your yes or no answer to me. The game will consist of trying to communicate with each other without speaking or writing, and without using our computers. After thinking about it for awhile, we realize that some sort of sign language or visual signal might work. But what kind of signal?

"Why don't I use the light in my office to signal whether I can go or not," you suggest. "If I can go, I'll turn the light on. If I can't go, I'll leave it off." We try out your idea, and it works perfectly. I'm able to see the light through your window as I drive by.

Now let's examine what we did. We used an electrical circuit to represent information. The wires, light bulb and wall switch in

your office constituted the circuit. But the electrical circuit, by itself, wouldn't have been enough. Equally important was our prearranged agreement about how I should interpret the light being on, or off. By agreeing to interpret it as we did, we were actually setting up a code. That code can be listed in the following table:

off = no
on = yes

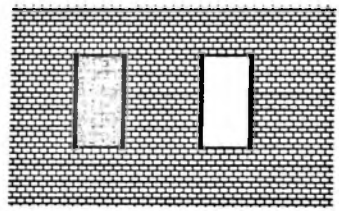
Now let's suppose that it's a week later, and we want to go to the movies again. This time, however, you have a scheduling problem that's a bit more complicated.

"Today is Wednesday," you tell me over the phone, "and I'll be working late at the office. How about going to the movies on Thursday, Friday, Saturday or Sunday? However, I won't know until tonight which one of those days I'll be free."

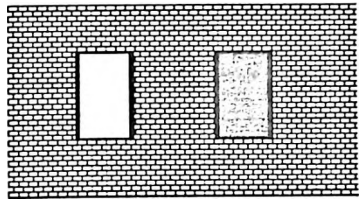
It's obvious that, for this communication, we're going to need more than just the single light bulb in your office. One bulb allowed us to signal only two messages (yes or no) whereas we now need to be able to communicate four messages (Thursday, Friday, Saturday or Sunday). We're going to need more than just the one circuit.

After a moment's thought, you come up with a solution. "How about using the light in my boss's office, in addition to the light in my office? Her office is right next to mine, and it, too, has a window that can be seen from the street." But will two light bulbs be enough to communicate four messages? To find out, let's see how many combinations can be created with two circuits.

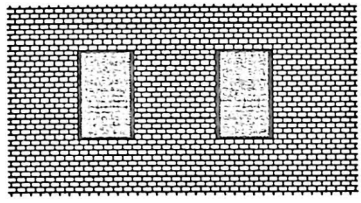
You could leave your light off and turn your boss's light on:



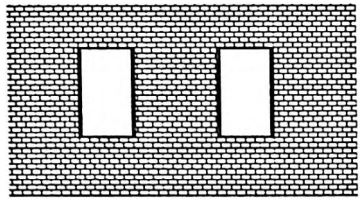
Alternately, you could leave your light on, but leave your boss's light off:



A third possibility would be to leave both lights off:



The last option would be to turn both lights on:



With this two-circuit system, we'll have to agree on a code, as we did before, that assigns each one of these four combinations to one of the days you might be free. Let's assign the days and list the code:

off on	=	Thursday
on off	=	Friday
off off	=	Saturday
on on	=	Sunday

This should work, and in fact, it does. When I drive by your

office, I see that your light is on and your boss's light is off. You've signaled that Friday is the night you'll be free.

We used two circuits this time, but what if we used three? "Yeah," you chime in. "I could use the light in the room that's on the other side of my boss's office." After doing some calculating, we discover that with three circuits we will have eight combinations.

Eight combinations will work just fine for the following week's signal, since you're not sure whether you'll be able to go to the movies at all that week. To accommodate this, we use the eight combinations to set up the following code:

off off off	=	you can go only on Sunday
off off on	=	you can go only on Monday
off on off	=	you can go only on Tuesday
off on on	=	you can go only on Wednesday
on off off	=	you can go only on Thursday
on off on	=	you can go only on Friday
on on off	=	you can go only on Saturday
on on on	=	you can't go any day of the week

You'll remember that our first experiment at communicating used one circuit, which allowed us to represent two alternatives. With two circuits, we had four alternatives, and three circuits has produced eight alternatives. As you can see, whenever another circuit is added, the number of alternatives that can be represented is doubled.

Using this formula, we can calculate how many alternatives can be produced by greater numbers of circuits:

4 circuits will produce 16 alternatives

5 circuits will produce 32 alternatives

6 circuits will produce 64 alternatives

7 circuits will produce 128 alternatives

8 circuits will produce 256 alternatives

It should come as no surprise that the signaling we have done -- using the lights in your office -- is analogous to what goes on inside a computer. In fact, if we assume that your office building has eight windows facing the street on each floor, we could make our analogy fit the ST quite well.

The reason your building should have eight windows per floor is that our computers deal with their millions of electrical circuits in groups of eight. In the same way that we used the three electric light circuits in your office to represent a day of the week, our STs can use a group of eight tiny electrical circuits to represent a character of the alphabet, such as the letter H.

Here's how that works: When your ST wants to store the letter H, it sets eight circuits in the following on/off pattern:

off - on - off - off - on - off - off - off

After storing the letter H, if the computer then wants to store another letter -- say the letter E -- it sets eight other circuits in a different on/off pattern to represent E. And by using additional groups of eight circuits, the computer can store enough letters to make up words, sentences and whole documents.

As you type a word on your ST -- for instance, the word HELLO -- each of the letters of the word is "stored" in the computer's memory chips. Applying our light bulb analogy again, let me use five floors of the office building (one floor for each letter of the word HELLO) and turn on (or off) the individual lights on these floors in such a way that the pattern is the same as the switch settings in that part of your ST where the word HELLO is stored. Figure 1.1 shows what your office building will look like from the street.

If this chapter is beginning to look to you like a programming manual, don't worry. We'll be finishing our discussion of switches very soon. And anyway, unless you become an advanced programmer, you're never going to have to use (or even understand) this on/off system. We're just setting the stage for future topics.

So far while writing this chapter I've used the words *off* and *on* to designate the condition of an electrical circuit. That's been all

right until now, but if we were to start dealing with more than just a few letters of the alphabet, the series of *offs* and *ons* that would be required would get unnecessarily confusing.

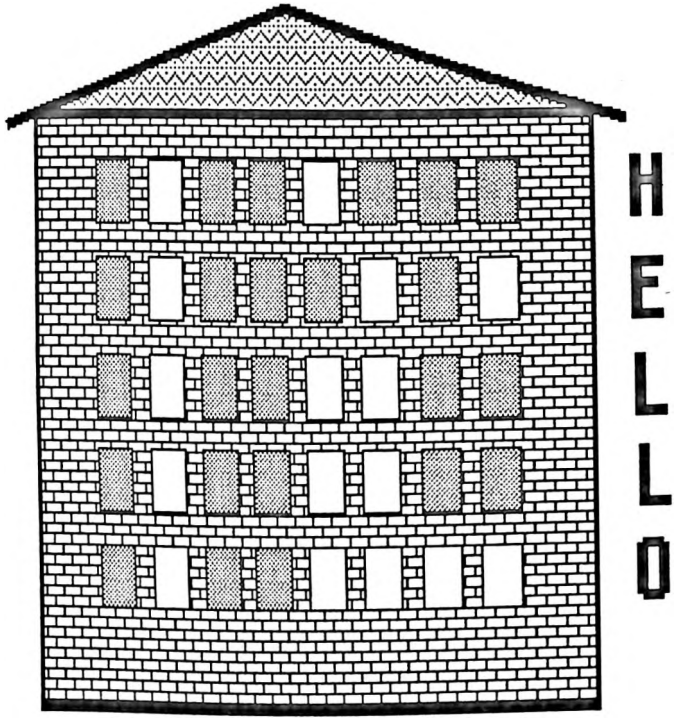


Figure 1.1

To avoid confusion, computer scientists find it easier to use the symbols 0 and 1 to represent *off* and *on*. In fact, you've probably seen these symbols already. Many computer components, such as your ST and possibly your printer, designate the positions of their power switches with an 0 (for *off*) and a 1 (for *on*).

Using 0s and 1s, here's how we would represent, on paper, the condition of your computer's eight circuits that are being used to

store the letter H:

01001000

These eight 0s and 1s tell us that the first three circuits in your computer are off (we read from the right to the left), the next circuit is on, the next two are off, the next is on and the last is off.

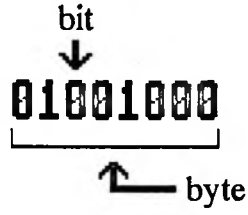
When 0s and 1s are strung together in this manner, it's called *binary code*. The word binary, meaning two (as in bicycle and bimonthly) refers to the two possible states of each electrical circuit: either on or off.

BITS AND BYTES

Before wrapping up this chapter, let's introduce a few more terms. When the symbols 0 and 1 are used to represent the on/off conditions of electrical circuits, the 0s and 1s are called *binary digits*. However, the phrase binary digit is usually shortened to the word *bit*. (*bi*, from binary, and *t*, from digit.)

The other term that we'll want to be familiar with is *byte* (pronounced "bite"). A byte is a group of eight 0s and 1s that have been arranged to represent one character of data (such as a letter of the alphabet, a digit, or a special character).

So, when you see a single 0 or a single 1, you're looking at a bit; and when you see eight 0s and 1s together, you're looking at a byte:



Let's return to your office building one last time before moving on to the next chapter. You already know that the reason I set you up in a building with eight windows on each floor was that I wanted the building to be analogous to your ST's memory chips. How tall do you think your building would have to be in order to store as much data as a 520ST can? The answer is 524,288 stories high!

This means the building would have over four million windows. (4,194,304, to be exact: 524,288 floors x 8 windows per floor). That's how many electrical circuits are contained in a 520ST's memory chips. A 1040ST is twice as big, so its memory chips have 8,388,608 circuits. A Mega 2 has 16,777,216 circuits and a Mega 4 has 33,554,432.

Let's move on to the next chapter where we'll deal with some topics that are related to binary: ASCII, hexadecimal, and decimal.

* * * *

ACKNOWLEDGMENTS

I would like to thank the following people for the help they provided during the research and writing of this book: Ralph M. and Virginia Turner, Mark Minato, Mickey Angell, Eb Arseneau, Sandy Austin, Todd Bane, John R. Beck, Dave Beckler, Jim Berry, Ron Bewick, R.C. Binder, Miles W. Brown, Alan Burnes, Mark Campbell, Mikel Cawthon, Harry A. Clough, Donald F. Crowhurst, Al Davis, Dr. Herbert H. Dobbs, Carl W. Flach, Don Geppert, Thomas Hansen, Mike Hendricksen, Marty Herzog, Chris Latham, Tim Laughrin, Arnie Lee, Gurdy Leete, Dr. Gerald Lemberg, Randy Lewis, Mike Lindsay, Ron Luks, Rod MacDonald, John Martin-Jones, Phil Michalson, George Miller, Earl Miller, Tom Murnane, Burke R. Oppenheimer, Sailesh Patry, Jarvis H. Post, M.D., Nathan Potechin, Gilles Raymond, Phil Reeves, Arie Rigby, Ron Ritzman, Brad Roltgen, Jonathan Sabin, Jason Saffer, David Sinn, Frank Sommers, Robert L. Spence, John Szasz, Ginny Teal, Don Terp, Craig Thom, LeRoy Valley, Rick Vega, Joe Waters, Sandy Wilson, Ed Zahniser, and Dr. Irving Zelkind.

Chapter Two

ASCII, HEXADECIMAL and DECIMAL

QUESTION: After entering a number of names and addresses in my data base manager, I used the program's report function to print out the records on my dot matrix printer. The first part of the printout went normally, but half way through the printing my Epson LX-80 printer stopped inserting spaces between the lines. (See Figure 2.1)

```
Sal, 226 Noble Ave., Babbitt, MI, 55706
Terry, 399 Union St., Albany, NY, 12801
Floyd, 5858 Rawlston, Ranger, TX, 76470
Bob, 1256 Baker Ave., Troy, NY, 12309
Sally, 356 Ulloa St., Chico, CA, 94111
Tom, 236 Grant, Gakona, AK, 99586
Sue, 567 A. Street, Mench, PA, 16226
Dorán, 1299 Root, Belmont, CA, 94002
```

Figure 2.1

Thinking that possibly the printer was at fault, I turned it off, then on again, then sent the report to the printer a second time. The same thing happened again. Perplexed, I used the data base management program to look at the record which corresponded to that part of the hardcopy where the change in printing occurred, thinking that maybe I could notice something strange about that record. But the record looked normal, so I tried SHOWing the file from the Desktop. It looked OK there, too. (See Figure 2.2). I'm sending you a disk with the file on it in the hope that you can tell me what's going on.

```
Sal, 226 Noble Ave., Babbitt, MI, 55706
Terry, 399 Union St., Albany, NY, 12801
Floyd, 5858 Rawlston, Ranger, TX, 76470
Bob, 1256 Baker Ave., Troy, NY, 12309
Sally, 356 Ulloa St., Chico, CA, 94111
Tom, 236 Grant, Gakona, AK, 99586
Sue, 567 A. Street, Hench, PA, 16226
Doran, 1299 Root, Belmont, CA, 94002
---END OF FILE---
```

Figure 2.2

ANSWER: You were on the right track in trying to find a suspicious entry in that record. But although you were looking in the correct place, the cause of your problem remained hidden from view.

Your problem was created by a special printer control character which somehow got embedded in that record. You weren't able to see the character, however, since control characters usually aren't displayed. That is, even if they are present in a file, they won't appear on the screen when a file is viewed from within most programs, such as word processors, data base managers, etc. Neither are control characters visible when you SHOW a file from

the Desktop.

In order to see control characters, you'll need to use a special program called a "file editor." But before we crank up one of these intriguing programs, let's make sure we're prepared. Using a file editor is a bit more complicated than merely holding a magnifying glass to your disk, so before we proceed we'll want to become familiar with a number of basic computer concepts.

PREREQUISITES FOR USING A FILE EDITOR

In order to locate and examine all the characters that make up a file, you need to know how to read the hexadecimal numbering system. We'll also want to understand why file editing programs (as well as programmers) rely on hexadecimal. And while we're at it, we'll learn something about decimal codes and the ASCII standard.

Hexadecimal (which is often referred to merely as "hex") and ASCII are actually very basic concepts. A knowledge of one or both of these concepts is often necessary if you want to access your printer's extended features, manipulate and recover data on your disks, and (in some cases) look for computer viruses. But, although basic, these concepts could also be considered advanced, because very few computer owners know much about them.

I realize that you may already be moaning at the prospect of plodding through explanations of such complicated-sounding subjects. But don't worry. We're going to be covering things slowly, very simply, and step by step.

To start, let's go back to an idea that was covered in the previous chapter. Remember how, prior to our first visit to the movies, you decided that leaving your light off would mean "no," and leaving it on would mean "yes"? This was purely an arbitrary choice. You could have just as easily chosen off to equal "yes" and on to equal "no."

This same sort of arbitrary decision has also been made regarding a computer's binary coding system. There's no valid reason, for instance, why the circuit combinations 01001000 should represent the letter H, other than convention.

The present system of representation came about a number of

years ago when a group of computer scientists agreed on a standard for representing numbers and characters in computers. This standard, called the American Standard Code for Information Interchange, is used in most computers today. It's usually referred to by its acronym, ASCII, which is pronounced "as'-key."

ASCII is actually a character code. By this we mean that it's an agreed-upon scheme for representing characters by combinations of "0s" and "1s." The Atari ST ASCII code uses eight bits (i.e., a series of eight 0s and 1s) to represent all the characters that appear on your keyboard: the numerals 0 through 9, the upper and lower case letters, and the punctuation marks. In addition to these familiar characters, there are 32 non-printing control characters, as well as 127 "extended" (additional) characters. In all, there are 256 characters in the Atari ST ASCII code.

We'll be learning more about ASCII in the pages that follow, but let's proceed to cover a few other things prior to actually using a file editor.

NOW FOR A TRUE STORY

When I first became interested in computers, all I knew about them was that they used the binary numbering system.

One day I decided to create a printer driver for an offbeat printer I had borrowed. After reading my word processor's manual, I quickly became confused. It instructed me to enter hexadecimal codes that would be sent to the printer. As examples of hexadecimal, the manual listed 1B,34 and 1B,0E. I knew nothing about hexadecimal, and after two days of frustrating attempts, I still hadn't created the printer driver. Remembering that W. C. Fields once said, "If at first you don't succeed, give up," I decided to switch to a different word processor.

I was pleased to discover that my new word processor didn't require hexadecimal code in order to create a printer driver. Instead, it asked for "escape sequences in decimal," such as 27,54 and 27,33,1. After two more days, during which I tried all kinds of decimal combinations, I still didn't have a printer driver. I was more confused than ever.

"What does this computer want, anyway?" I cried. "Does it use binary code sometimes, hexadecimal code other times, and decimal code on even other occasions? Or is it the *programs* that can't agree on a standard? Does one word processor send hexadecimal signals to the printer, and another word processor send decimal signals?" It wasn't until many months later that I discovered why these different numbering systems are used, and how they each fit into the picture.

Internally, your ST uses binary code exclusively, from the moment you press one of the keys on your keyboard. As soon as you press a key, a string of off/on signals is generated. These bits are then stored in the computer's memory, and they remain in this binary form regardless of what you tell your computer to do with them. For instance, if you send a file to the printer, the signals go in binary form. If you send a string of words to your modem, they go out the RS232 port in binary. And if you save a file to a floppy disk, the signals go to the drive in binary.

Why, then, are hexadecimal and decimal ever used? The answer is that, although the binary code's on/off, two-state nature is perfect for use in electrical circuits, binary code is not easy to read on paper. To prove this point, let's have you take a little test. See if you can quickly determine whether the following groups of 0s and 1s are the same.

01001011; 01001011; 01001011; 01001011; 01001011;
01001011; 01010011; 01001011.

How did you do? Did you discover that all the groups are the same except the seventh group? It wasn't easy, was it?

HEXADECIMAL

To avoid the difficulty of dealing with long strings of 0s and 1s, programmers often use the hexadecimal numbering system. Hex numbers are considerably easier to recognize; and to illustrate this, let's give you the test again. But this time, the same information will be displayed in its hexadecimal form:

4B; 4B; 4B; 4B; 4B; 4B; 53; 4B.

This time it was a lot easier to see that all the groups were the same except the seventh group. But although hex is easier to read than binary, the average user may find hex a little strange at first. Because of this, I'll be explaining only those things about hex that you really have to know.

At the back of this book you'll find appendices A, B and C, which contain decimal-hex-ASCII conversion tables. With these tables you'll be able to convert from other numbering systems into hexadecimal, and vice versa. Because the tables are so easy to use, you won't need to be concerned with the mathematical aspects of hex.

However, there are a few things about hex that you might want to know. First, hex is a base 16 numbering system. By that is meant that the system uses 16 symbols. This contrasts, of course, with the decimal system which we learned in grade school and which we use every day. Our decimal system is called a base 10 numbering system, since it's built on the use of the 10 symbols 0 through 9. The binary system, which was discussed in the previous chapter, is a base 2 system and uses the two symbols 0 and 1.

The 16 symbols of the hexadecimal system are

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Here's how you would count up to 30 in hexadecimal:

1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12,
13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E

Pretty weird, isn't it? But remember, you're not going to have to understand hexadecimal, nor are you going to have to count in it. All you'll be doing is using the tables in the appendices to convert a number (or a character) from one numbering system into its

hexadecimal equivalent. Let me show you how easy that's going to be.

Below is a portion of the Decimal-Hex-ASCII table from Appendix B. Using this table, you can easily determine that the equivalent of decimal 59 is hex 3B. You'll also be able to do conversions in the opposite direction: you can see that the equivalent of hex 40 is decimal 64.

Decimal		Hexadecimal
56	=	38
57	=	39
58	=	3A
59	=	3B
60	=	3C
61	=	3D
62	=	3E
63	=	3F
64	=	40
65	=	41

Although we haven't answered the question that was posed at the beginning of the chapter, we eventually will. But now it's time for some fun. Let's move on to the the next chapter where we'll be using a file editor to start playing with hex.

Chapter Three

FILE EDITORS and CONTROL CODES

A file editor is a program that allows you to view and edit files. Unlike word processors, however, which display only some of the characters that make up a file, file editors allow you to see *all* of a file's contents.

There are a number of file editing programs for the ST, some of them in the public domain, others commercial. I'll be mentioning three of them in this book, but let's start out with one of the free ones, TINY TOOL. TINY TOOL, which works in high or medium resolution, is a desk accessory, so you'll want to have it installed on your boot disk when you turn on your computer. If you don't know what a desk accessory is, or you have any problems installing TINY TOOL, refer to Chapter 8 of my *Atari ST Book*.

Although a file editor can be used to examine any kind of file you want, we'll deliberately choose a text file rather than a program file. A text file will contain characters that are recognizable as English words, whereas a program probably wouldn't. This will make the exercise easier.

Before you call up the TINY TOOL program, here are two important things to know. First, if you're going to examine a file that's at all valuable, start out by making a copy of the file. Then use TINY TOOL to examine the copy, not the original.

Here's why you don't want to use a file editor on an important file. File editors allow you to do more than just *examine* files. They also allow you to *change* files. As a novice, it'd be very easy to inadvertently make a change in a file that would prevent that file from being used again. However, if you only work on a copy, any mistake will ruin only the copy, not the original.

The second warning has to do with hard drives. It will be good, at least for now, *not* to use the editor to examine a file that's on a hard disk. This is true whether or not the file is important. If you mistakenly alter certain bytes on your hard drive, the consequences could be disastrous.

Assuming you've made a backup of your target file, it's now time to use TINY TOOL. When we choose TINY TOOL from the Desk Accessory pull-down menu, we're presented with the initial TINY TOOL window. (See Figure 3.1)

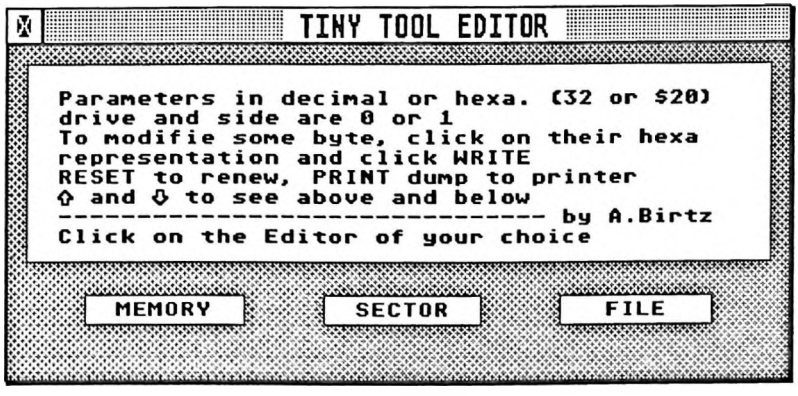

TINY TOOL EDITOR
Parameters in decimal or hexa. (32 or $20)
drive and side are 0 or 1
To modifie some byte, click on their hexa
representation and click WRITE
RESET to renew, PRINT dump to printer
⇧ and ⇩ to see above and below
---------- by A.Birtz
Click on the Editor of your choice
MEMORY
SECTOR
FILE

Figure 3.1

The three boxes at the bottom of the window allow you to choose what it is you want to view. We'll use the mouse to click on the "File" box, since we want to view a file. (We *don't* want to click on the "Memory" or "Sector" boxes.)

The familiar Item Selector box pops up, asking us which file we want to view. I'll use the selector to choose a text file which I created a few days ago in a word processor. After selecting the file, we're presented with the second TINY TOOL window. (See Figure

3.2)

Press the [Return] key and the large rectangular white box in the window will fill with numbers and letters. There are actually three groups of numbers and letters in the white box, each consisting of different information.

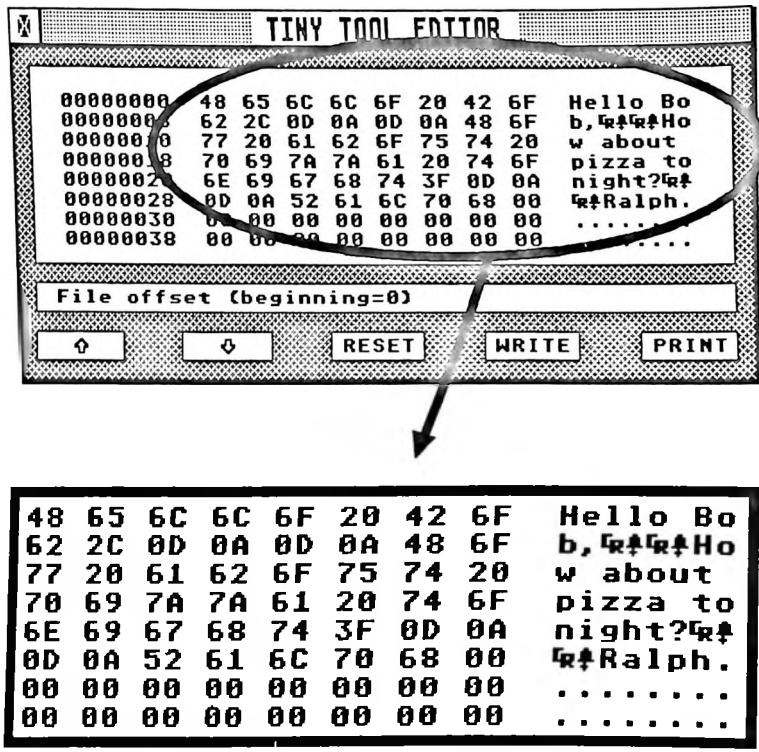


Figure 3.2

The group on the right comprises an ASCII display of the beginning of the file. Only the first 64 bytes (i.e., characters) of the file are displayed. If we want to look at characters that are located further on in the file, we can use the mouse to click on the [Down] arrow in the lower left hand portion of the TINY TOOL window. The [Up] arrow will move us backwards in the file.

The group of letters and numbers in the middle of the window consists of hexadecimal equivalents of the ASCII characters in the right group. To understand the relationship between the middle and right groups, let's consider the first character in the file, the ASCII character H. If we look at the top line of the hex group, we'll see that the number at the far left is 48. This makes sense, since ASCII H is the same as hex 48. (You can verify this by referring to the Decimal-hex-ASCII Table in Appendix B) Similarly, we see that the second character in the file, ASCII e, is hex 65.

Just so it's clear to you what the relationship between hex and ASCII is, I'm going to use an analogy involving French and English words. Below, I've created a chart with French words on the left and the equivalent English words on the right:

chaise	chair
nuage	cloud
fille	daughter

The English word "chair" means the same thing as the French word "chaise." These words are just expressed in different languages.

The TINY TOOL window is similar to this. ASCII is on the right, hex is in the middle. The ASCII on the right is a true representation of the data that's in the file, but the hex in the middle is also a true representation of that same data. We could say (very loosely) that hex is just a different language than ASCII. In the same way that chair and chaise are just different but equally correct words to describe a piece of furniture you sit on, ASCII H and hex 48 are just different ways of describing the first byte in this file.

QUESTION: I see a number of strange characters on the second line of the ASCII display. One looks like the letters CR and the other looks like a bell. (See Figure 3.3.) What are these symbols?

ANSWER: These are control characters. The $^{C}_{R}$ symbol is called the carriage return and the bell symbol represents a line feed. If you've never used a file editor, you probably haven't seen these characters before.

To understand these (and other) control characters, let's turn away from the TINY TOOL program for the moment and think about how your ST tells its printer to print a page of text.

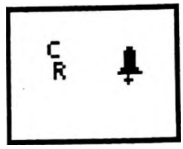


Figure 3.3

When you create a document with your word processor and send it to the printer, there are two types of signals that go out the printer port: those representing the letters and numbers you typed, and those representing special instructions for the printer. The letters and numbers tell the printer *what* to print, and the special instructions tell it *how* it should print. You're aware of what the letters and the numbers are that are being sent to the printer, since they're visible on the printout. But you may not have been aware that special instructions were also being sent to the printer along with the text.

Your computer communicates these special instructions by sending control characters to the printer. The control characters tell the printer such things as where the ends of the lines are, when to tab, and when to skip to the top of a new page.

THE "HIDDEN" ASCII CHARACTERS

All of the characters, numbers and symbols on your keyboard can be found in the 256-character ASCII code. (See Appendices A, B and C.) The ten digits start at decimal 48, the letters of the alphabet start at decimal 65, and the remaining keyboard characters are scattered here and there between decimal 33 and decimal 127. But look at the 32 characters in Appendix A's table, codes 0 through 31. These 32 characters are all control codes. Unlike codes 32 to 126, however, the 32 control codes don't appear on your keyboard.

Let's investigate the role that control codes play in printing a document. To do this, we'll examine that short letter to Bob that

appears in Figure 3.2. Here's what I typed at my keyboard to create that letter:

Hello Bob,[Return]
[Return]
How about pizza tonight?[Return]
[Return]
Ralph

The word "Return", enclosed in square brackets, indicates a press of the [Return] key. Pressing [Return], of course, is what we do when we want to start typing a new line. It's also what we do when we want to skip a line.

When we press the [Return] key from within most word processors, two control characters get embedded into the text. One is the carriage return character. (This character is often referred to in shorthand as "CR". Its hexadecimal representation is 0D, its decimal representation is 13.) The other character that's created when you press [Return] is the line feed character. (This character is sometimes referred to as "LF". Its hexadecimal representation is 0A, its decimal representation is 10.)

Here's what happens when the text of this short file gets sent to the printer. First, the printer prints

Hello Bob,

Then the printer receives the carriage return control character. This causes the print head of the printer to return to the left-hand side of the platen. Next, the printer receives the line feed control character. This causes the platen to rotate, which feeds the paper up one line.

Obviously, the CR and LF characters are very important. But, although important, you normally don't see them. They don't appear on the screen from within most word processors, they don't appear on the printout from the printer, and they don't appear on

the screen when you SHOW the file from the Desktop. Nevertheless, these characters are present in the file, and a file editor will display them.

Other control characters that you might see if you use a file editor to examine a text file are the backspace character (BS; hex 08; decimal 8), the tab (HT; hex 09; decimal 9), and the form feed (FF; hex 0C; decimal 12). The form feed is used to skip to a new page.

Although I've concentrated on the use of control characters in sending instructions to a printer, these codes are also used by your computer in other situations, such as when special signals are needed to be sent to another computer over a phone line.

EMBEDDING CONTROL CODES INTO TEXT

There generally are two ways that control characters get entered to a file. We've discussed the first way already: when you press some of the keyboard keys - such as the [Return] key - the corresponding control characters are entered.

But in some programs there may be a way to selectively insert these characters into a file. Doing this is often referred to as "embedding control codes into text." Some programs don't allow this, others do. At the end of this chapter I discuss how to access control characters from within a number of popular programs.

Although my emphasis here is on word processors, other types of programs may also allow control codes to be entered. Data base managers, text editors, spreadsheets, and telecommunications programs are other types of programs that come to mind.

QUESTION: Why would you want to enter control characters into text? It sounds like the computer takes care of it for you by sending the proper codes when you press [Return], [Backspace], [Tab] etc.

ANSWER: [Return], [Backspace] and [Tab] represent only three of the 32 control characters. Many of the remaining characters can be entered only by embedding them selectively.

Let's consider the use of your printer. If you learn how to embed

control characters, a whole new world of printing options will open up to you. The manual for your printer, which up until then probably seemed almost useless, may finally get some use.

To illustrate this, let's consider the manual for my Panasonic KX-P1092 printer. The manual's table of contents lists over 50 special options that can be accessed with the printer, including pica or elite pitch, an international character set, double width, and compressed printing. Let's see what the last two modes look like when compared to regular text:

```
regular printing

this is an example of compressed printing

Double Width Printing
```

If I didn't know how to embed control codes, I'd be restricted to not more than five of my Panasonic's special features: underlining, italics, bold, and possibly superscript and subscript, depending on the particular word processor (or other type of program) I was using.

Moreover, there are some programs in which the *only* way to get even simple features - such as underlining or italics - is by knowing how to embed control codes. My data base manager is such a program.

QUESTION: But what do you *do*, specifically, to enter a control code?

ANSWER: That depends on the particular program you're using. As has been mentioned, many programs simply do not provide a way to embed control characters. But even among those programs that do provide for control code insertion, there is no standardized way of accomplishing the embedding.

Here's an example. On some printers, the SHIFT OUT code is what turns on double width printing. To send the SHIFT OUT code

to your printer from one word processor, you might have to key in the decimal code for SHIFT OUT, which is 14.

The fact that you have to enter the code in decimal is a requirement of the word processor, not a requirement of the printer. Once the word processor receives the decimal code from you, it "converts" it into binary code, and it's in this binary form that the signal gets sent to the printer.

Let's assume, however, that you switch to a program (such as dBMAN, from VersaSoft) which requires that control codes be entered with the CHR function. In dBMAN, you would type

```
CHR(14)
```

If you were using ST BASIC, you would type

```
CHR$(14)
```

Let's assume, however, that you switch to a program that requires control codes to be entered as control-key combinations. By referring to Appendix A, you can determine that typing [Control]+[N] will produce the SHIFT OUT signal. Again, it's the *program* that's requiring the pressing of [Control]+[N]. From the printer's standpoint, the print instructions from all these programs are the same.

USING THE CONTROL KEY

QUESTION: I don't understand what you mean by "typing [Control]+[N]."

ANSWER: This means that you should press and continue to hold down the [Control] key and then press the [N] key. This is done in the same way that you get a capital letter by using the [Shift] key.

In other books and magazines, control-key combinations are

often indicated differently. Sometimes you'll see "Control-N" or "CNTRL-N", or (as in Appendix A,) "^N". This latter example does *not* mean that you press the carat key. (The carat symbol looks like an inverted "V", and is produced by pressing the shift key in conjunction with the "6" key.) It's just that, when writing instructions, many people use the carat symbol instead of spelling out the word "Control."

If your program allows for the use of control-key combinations, you can refer to Appendix A in order to figure out what key combinations to press in order to embed a specific code. This table lists the decimal and hexadecimal codes for the first 32 places in the ASCII character set, as well as the control-key combinations that produce these characters.

QUESTION: In the data base management program dBMAN, pressing the [Control]+[W] combination saves the work in progress. What control code is that?

ANSWER: It isn't one of the 32 ASCII control codes at all. dBMAN, like many programs, uses control-key combinations to access the program's features. In similar fashion, Microsoft Write uses the [Control]+[S] combination to SAVE a document. But neither of these combinations have anything to do with embedding control codes into text.

THE ORIGINAL QUESTION

Now let's return to the question that opened the previous chapter. Why did your printer stop printing spaces between the lines? To answer this, let's use TINY TOOL to examine that file. First, though, let's make a copy of the original file so we can refer to it later.

Once we load the copy of the file into TINY TOOL, we'll have to mouse-click on the [Down] arrow to move through the file until we get to the suspect record.

Figure 3.4 shows us what TINY TOOL displays once we've moved to the proper place in the file. Everything looks normal *except* that EC symbol. (3rd line of the ASCII group, 5th character.)

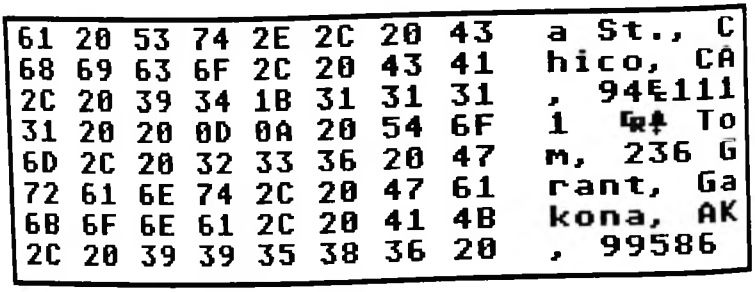


Figure 3.4

EC is a character we haven't encountered yet. The EC symbol is a graphic image for the escape character, which you may see referred to in books as "ESC".

Looking at the middle group of values in TINY TOOL, we see that the escape character has a hexadecimal value of 1B. We'll discuss the escape character shortly, but for now, take my word for it: the escape character should *not* be at this place in this file. It's gotten here somehow, although I'm not sure how. In any event, it's got to go, so let's remove it and see if that solves the printing problem.

MODIFYING A BYTE

File editors allow us to change parts of a file, one byte at a time. To do this from within TINY TOOL, we first mouse-click inside the hex group on the byte we want changed. In this case, this means mouse-clicking on hex 1B, which is on the third line.

As soon as we mouse-click, a new prompt will appear in the long rectangular box near the bottom of the window. The prompt tells us what the present value of our byte is (1B), and asks for the new value. We now have to decide what character to use as a replacement. Let's insert a space in place of the escape character. (The space is decimal 32, hex 20.)

TINY TOOL allows us to type in a new value in either decimal or hex. If we prefer decimal, we simply type the decimal code for a space, which is 32, then press [Return]. If we now look at the

ASCII group, we'll see that the escape symbol has been replaced by a blank space. (See Figure 3.5)

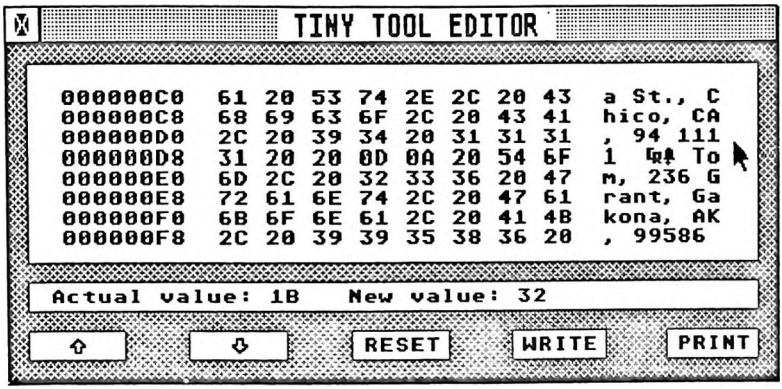


Figure 3.5

If we had preferred to type in the new value as a hex number, we would have typed

$20

then pressed [Return]. (The dollar sign is a common way of designating that what follows is a hex number.)

As mentioned earlier, the decimal notation and the hexadecimal notation for any given character are actually the same code, and they both have the same numerical value. For example, hex 4B has a numerical value of seventy five, and decimal 75 also has a numerical value of seventy five. (Hex 4B and dec 75 are the values for the ASCII character K.)

Although we've replaced the escape code with a space, this has only taken place in the ST's memory. We now need to make this change in the saved file. To do this, mouse-click on the WRITE box.

Now it's time to print a report from this newly altered file and see what happens. Hey, it works! The whole report is printed with

the proper spacing between the lines. (Actually, we'll need to perform one more step. Although we've solved the spacing problem by replacing the escape character with a space, this causes our output to become "...Chico, CA, 94 1111". We'll need to go back into our database manager and remove the space that's between the "4" and the "1".)

MULTI-BYTE CONTROL CODES

QUESTION: How did the presence of that escape character cause the printing problem?

ANSWER: In order to answer that, let's first delve into control codes a bit deeper. Remember I said that my Panasonic printer had over 50 special features that could be made use of?

Since there are only 32 control codes, some of which have already been assigned regular tasks such as designating a carriage return, there has to be a way to combine control codes so that more than 32 different printer instructions can be generated.

Here's how that's done. There are actually two types of control codes: *single-byte* control codes and *multi-byte* control codes. When a character with an ASCII value from 0 to 31 is used by itself, it's called a single-byte control code.

Contrasted with this, whenever an instruction consists of the escape code *followed* by an additional instruction, the sequence is called a multi-byte control code. (The escape code is decimal 27, or hex 1B.) As their name suggests, multi-byte control codes consist of more than one byte.

Single-byte codes never include the escape code. Here are two examples of single-byte codes, together with the results they produce on my Panasonic:

decimal 7 = sounds a bell (or buzzer)
decimal 15 = sets compressed printing

When you first start experimenting with control codes, it might

simplify things if you try some single-byte codes and get them working properly before tackling multi-byte codes.

In multi-byte control codes, the escape code is used as a prefix, letting the printer know that what follows is an instruction. Here's an example. In order to turn on elite pitch printing on my Panasonic, I should send (in hex)

1B,4D

1B is the hexadecimal value for the escape code, and its purpose is to tell the printer, "What comes next is an instruction, not a character to be printed."

If, instead, I had sent to the printer just the letters 4D (in hex) without the 1B, the printer would merely print the letter "M", since 4D hex = M. But by prefixing my command with 1B, the printer is commanded to treat what follows (the 4D) as an instruction. And in the case of my Panasonic, the 1B,4D instruction will turn on elite pitch printing. (With another brand or model of printer, the 1B,4D instruction might produce a different result.)

You'll also run across multi-byte control codes that consist of more than two bytes. Here's an example:

1B,51,n

My Panasonic printer responds to this control sequence by setting the position of the right margin to whatever value is assigned to n. The 1B is the escape code, the 51 is called the *lead-in*, and the n is the *parameter.*

n is a variable byte value indicating the number of characters that can be printed before reaching the right margin. For instance, if we send 1B,51,19 (in hex) to the printer, then print a string of 40 numbers, this is how those 40 numbers will be printed:

1234567890123456789012345
678901234567890

As you can see, although the string was 40 characters long, only 25 characters were permitted on the first line, since we had assigned hex 19 (which has a numerical value of 25) to n. The remaining 15 characters of the string had to be on the second line.

If, however, we send 1B,51,23 (in hex) to the printer, then print the same 40 numbers, the following will be printed:

```
1234567890123456789012345678901234567
67890
```

In this case, since hex 23 has a numerical value of 35, the right margin is set so that only 35 characters can be printed on a line.

Multi-byte control codes are often referred to as *escape sequences*, since they are sequences which begin with the escape code. Note, however, that although I've used hexadecimal code in the above examples, the same sequences could also be entered in decimal form from within a program.

For example, the hex sequence 1B,51,23 could be entered as decimal 27,81,35. Of course, whether you enter the sequence in hex or decimal will usually be determined by the requirements of the program you're using.

Our knowledge of escape codes is now sufficient to deal with your last question: how did the presence of the escape character in your file cause the printing problem? To answer this, let's look at Figure 3.4 again. The ASCII display shows us that what followed the escape character were the numbers 1111.

I'll look in an Epson LX-80 manual and see if "escape 1111" (or a part of "escape 1111") is the code for any of the printer's features. Ah ha! "Escape 1" is the code to set line spacing to 7/72 inch. (The default for the printer is 1/6 inch.) When your printer received the "escape 1" escape sequence, it changed its line spacing. We now know what caused your problem, and why. Solving the mystery wouldn't have been possible, though, without a file editor.

One final note regarding the escape code. This code should not be confused with the ST's [Esc] key, which does not produce the 1B escape code (in most situations). What it is that the [Esc] key *does*

do will vary depending on what program you're in.

In dBMAN, for instance, [Esc] cancels a command. In the Degas paint program, on the other hand, it's used to define a block. So how *do* you enter the escape code? Just like any other control code. Depending on the program you're in, you may have to enter decimal 27, hexadecimal 1B, or type Control+[. (Refer to Appendix A.)

MISCELLANEOUS NOTES

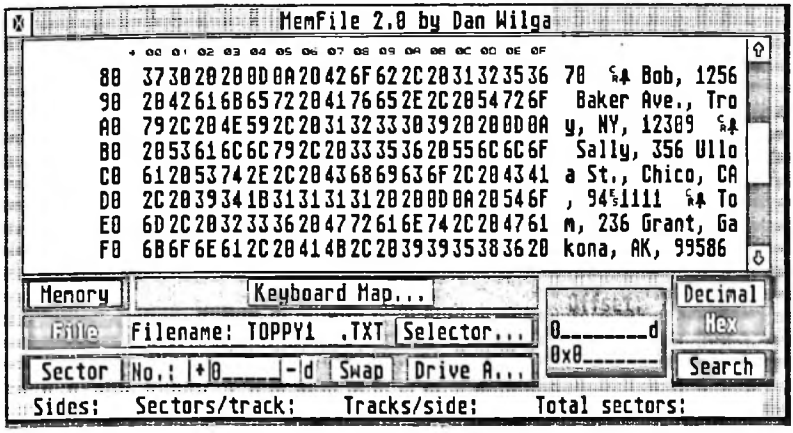


Figure 3.6

1. MEMFILE is another public domain file, sector, and memory editor. (See Figure 3.6) This program's search feature allows you to quickly locate an ASCII or hex character anywhere in a file.

2. The "Atari ST ASCII Character Set" has 256 characters in it (decimal codes 0-256). However, only the first 128 characters of this ST character set should be considered "true ASCII characters." Those characters with decimal codes from 127 through 255 are often referred to as the "extended ASCII characters." The ST's extended character set consists of foreign characters, as well as a number of special characters and symbols. (See Appendix C.)

3. A number of programs for the ST allow insertion of control codes into a document. One such program is Word Writer, from Timeworks. If I want my Panasonic KX-P1092 dot matrix printer to print in double width, I have to send to the printer (in decimal) 27,14. From within Word Writer, I first have to key in

P/27,14

Then I press the [F6] function key. The "P/27,14" will disappear and I can type in the text that I want to be double width.

From within the Regent Word word processor, the way in which I embed the code is slightly different. To turn on double width printing for my Panasonic printer, I type

[Control]+[X]27[Control]+[X]14

Note that with Regent Word, I must use the [Control]+[X] command not only at the beginning of a control code, but also between bytes of a multi-byte control code.

From within VIP Professional (a Lotus 1-2-3 clone for the Atari ST) I key in the commands "/PPOS", then enter the following characters:

\027\014

ST Writer, the very popular public domain word processor, is also capable of embedding control characters, as well as characters from the extended character set. To insert control characters (i.e., anything with a value between 0 and 31) you type [Control]+[X] before and after the control code that you want to embed. Thus, to turn on double width printing on my Panasonic, I first have to refer to Appendix A at the back of this book, where I can ascertain that [Control]+[N] is the Control-key combination that produces

decimal 14. Then from within ST Writer I type

[Control]+[X][Control]+[N][Control]+[X]

(Note, however, that ST Writer deviates in a few instances from the standard Control-key combinations that are presented in Appendix A. Refer to the tutorial for ST Writer.)

To print characters from the extended ASCII character set with ST Writer, you first should print out a copy of the file "STWCODES.DOC," which accompanies the program. You'll need to refer to this text file in order to determine which keys to press to obtain specific extended characters. Typing [Alternate]+[X] puts ST Writer into its "alternate character mode." In this mode, each key will produce a different extended character. For instance, pressing the comma key produces the "¼" symbol. Not only are the extended characters entered into the document that you're editing, but the characters are also displayed on the screen. However, whether or not what you see on the screen will be printed by your printer will depend on the type of printer you have.

4. There may be times when you'll want to use control characters to instruct your printer to select a particular font. For instance, let's say that you want to use WordPerfect (the word processor) to print the copyright symbol (the letter "c" inside a complete circle) on a Hewlett Packard Deskjet printer.

Whenever you want to print a character or symbol that's not on your keyboard, you first have to determine whether your printer is capable of printing the character you're interested in. To do this, look at your printer's manual.

Page 8-6 of the Deskjet manual lists 256 characters in the "PC-8 character set." Although other character sets also reside in the printer's ROM chips, the PC-8 set is the default set. (Other printers may similarly have a number of different character sets available.) If we look at the PC-8 character set on page 8-6, we find that the copyright symbol isn't available. However, page 8-7 displays the "Latin 1" character set, which *does* contain the copyright symbol. But before we can send the copyright command from WordPerfect, we've got to send a command that will cause the printer to switch

from its default PC-8 character set to the Latin 1 character set.

Page 4-12 tells us that "Esc (0N" will cause the Latin 1 set to become the primary character set. To find out what command to send to print the copyright symbol we look at page 8-7, which displays all the characters of the Latin 1 set. All the codes are listed in hexadecimal.

The table has hex numbers along the top, as well as along the left side. To find out what the hex code for the copyright symbol is, (a) locate the copyright symbol, then run your finger straight up until you reach the top line of hex numbers. You'll find A0 at the top. (b) Now look at the copyright symbol again, then run your finger straight to the left until you get to the vertical column of hex numbers. You'll find the digit 9. Adding 9 to A0, we get A9. This tells us that the hex number A9 is the code that will print the copyright symbol.

However, we've got a conversion to make. Whereas the Deskjet's manual gives us a code in hex, WordPerfect needs its codes in decimal. Refer to Appendix C at the back of *Atari ST Subjects*, where you'll see that hex A9 is equal to decimal 169. With this information, we can now go into WordPerfect and enter these codes.

To embed control codes from within WordPerfect, refer to page 460 of the WordPerfect manual in the section called "Printer Commands." We first need to send "Esc (0N" to switch to the Latin set. Referring to Appendix A, we find that "Esc" is equal to decimal 27; "(" is decimal 40; "0" is decimal 48; and "N" is decimal 78. Therefore, from within a document window press [Control]+[F-8], then press [6], then enter ‹27›‹40›‹48›‹78› then press [Return] twice. (From within WordPerfect, each decimal code should be surrounded by angle brackets.) To enter the code for the copyright symbol, we enter [Control]+[F-8], then press 6, then enter ‹169› then press [Return] twice.

(Note that, instead of entering ‹27›‹40›‹48›‹78›, WordPerfect will allow you to enter ‹27›(0N. However, since some of you have never dealt with control codes before, I felt that it might be less confusing for you to be told to change *all* four of the characters in the "Esc (0N" sequence into decimal, rather than just the first character.)

Chapter Four

ASCII FILES

Let me tell you about the most embarrassing computer experience I've ever had.

Soon after I started using a computer, I was hired by a publishing company to write a chapter for a book. The editor and I didn't own the same type of computer, nor did I have a modem, so after I had finished the chapter I brought my computer to his office. Our plan was to transfer the file from my computer directly to his through a null-modem cable. (A null-modem cable is a cable used to connect two computers together for the purpose of transferring information or files between the computers. It connects to the computers in the same way that modems do [i.e., to the RS-232 ports] but since the cable accomplishes the transfer of information without the use of a modem, the cable is called a null-modem cable, i.e., a "no-modem" cable.)

The editor hooked up the cable and I started to fumble with my computer's telecommunications program, which I had never used before. Once we both were ready, I sent the file to his computer.

"Hey, what'd you do?" he said, after loading the file into his word processor. The file that he received, instead of consisting of readable English words, was composed of a bunch of garbage characters, blank spaces, and skipped lines.

Embarrassed, I loaded the master copy of the file from my computer's floppy disk into my word processor to see if, somehow, the file had become corrupted before I had transferred it. But the file was fine, and consisted of readable English words. We tried sending it again, but the same thing happened.

Neither of us knew what had gone wrong or what to do, so I

called a friend of mine who was the head of computer operations at a large corporation. After describing the problem, my friend said, "Did you save the file in ASCII format before you sent it?"

"What's ASCII format?" I said. There was a long pause at the other end of the line, and I thought I heard a slight moan.

"Listen, Turner, have you ever seen a file with a .TXT extender?"

"Sure," I said, glad that I could admit to knowing *something.*

"Well, that's an ASCII file. In order for it to be readable by another computer, the file has to be in ASCII format before you send it."

After I thanked him and hung up, I changed the name of my file from BOOK.DOC to BOOK.TXT, then sent the file a third time. As before, though, all that arrived were garbage characters. Feeling even more like a fool but not knowing what else to do, I called my friend again and told him what had happened.

"Oh, come on, Turner! Merely changing the *extender* of a file doesn't turn it into an ASCII file! You've got to save the file in *ASCII format* from within your word processor."

After figuring out how to do this, I transmitted the file again. This time the process went without a hitch, and the file was readable and usable from within my editor's word processor.

WHAT AN ASCII FILE IS

There are many different types of file formats. Because most programs, such as WordPerfect, Word Writer, or dBMAN, have their own special way of recording data in a file, it is often impossible for one program to use a data file that was created by another program.

However, there is a universal type of file format that makes possible the transfer of data files between different types of programs, and between different types of computers. Files that are in this format are called *ASCII files, ASCII text files* or *text files,* and these files consist of text data in a very simple, streamlined format.

One characteristic of an ASCII file is its ability to be read and understood when viewed from the Atari ST's Desktop. (To do this,

double-click on the file's name or icon, then choose "SHOW" from the dialog box that appears.) Files that aren't in ASCII format often aren't understandable when viewed from the Desktop. For example, if you save a file in Regent Word's "native format" and then SHOW it to the Desktop, you'll see only two or three characters at the top of the screen, regardless of how long the file is. Luckily, many programs, including Regent Word, have the capability of saving a file in either ASCII or native format.

THE DIFFERENCE BETWEEN ASCII AND NATIVE FORMAT

ASCII files contain a minimum amount of information, and in this sense they're "bare files." Although they contain all of the necessary text characters (such as letters of the alphabet, numbers, punctuation marks, spaces, etc.) an ASCII file will contain only the minimum number of control characters. The control characters in an ASCII file will record information about such things as where a line will end, where a new paragraph will start, etc.

Contrasted with this, a file that is in native format will often contain a number of additional control characters, as well as all the text and control characters contained in an ASCII file. For instance, a word processing file that's saved in native format might contain special control characters that designate which words should be italicized, underlined, or be in boldface type. Other control characters might designate where the left and right margins should be. (See Chapter 3 for more on control characters.) Native format files are often in a coded form that makes sense only to the program that generated the file.

If your word processor is capable of saving in both native and ASCII format, you might want to conduct a test. Compose a file and save it in native format. Then save the same file in ASCII format. Now go to the Desktop and read the size of each file, in bytes. (To do this, mouse-click once on the file's name or icon, then choose SHOW INFO from the FILE Menu Bar.) You'll probably find that the native file is slightly larger than its ASCII counterpart, even though they both are the "same file." This is because the native format file contains more formatting

information than the ASCII file.

WHICH IS BETTER: ASCII OR NATIVE

If a program has the ability to save files in both ASCII and native format, which option should you choose? The answer will depend on what you want to do with the file. Unless you know that you'll be using the file in a situation that will require ASCII format, you should save it in native format. Here's why:

The fact that the program's creators have gone to the trouble of providing a native format indicates that this format is capable of containing information that couldn't be stored in a file saved in ASCII format. In fact, in most programs that employ a native file format, you're going to have to save the file in that format in order to make use of (and save) all of that program's features.

WHEN ONLY ASCII WILL DO

Although most of the time you'll want to save a file in native format, there nevertheless will be situations where your file will *have* to be in ASCII format:

1. FILE IMPORTING AND EXPORTING. This situation involves taking a file that has been created in one program and loading it into another program. Here's an example:

   Let's say you've used Word Writer to compose a magazine article, but you now discover that the magazine wants it submitted in Microsoft Write format.

   Since the two programs have their own native formats (which aren't compatible with each other) you can't open Microsoft Write and load in the story, since it has been saved in Word Writer's native format.

   Instead, open Word Writer and load in the story. Then send a copy of the story to your printer. The copy will allow you to see which words are italicized, underlined, etc. Next, save the file in ASCII format from within Word Writer.

Now open Microsoft Write and load in the ASCII file. This ASCII file no longer contains italics, underlining, etc., so look at the printout you made, then add these special features by using the appropriate Microsoft Write commands. Finally, save the document in Microsoft Write's native format.

Note: there are two ways to deal with the fact that Microsoft Write likes files that are being imported into it to have a .DOC three-letter extension.

Alternative 1: When you are still in Word Writer and you're ready to save the file in ASCII format, give the file a .DOC extension, while changing the filename proper (so as to distinguish it from the original, native file).

Alternative 2: In Microsoft Write, when you choose "Open," and the Item Selector pops up, replace the letters "DOC" on the Directory Line with "*.*". This will cause the ASCII file to be displayed in the Item Selector's Directory Window, regardless of what the file's extension is.

2. MODEM TRANSFER OF FILES. When you create a file that will be transferred through a modem (or a null-modem) to a commercial information service (such as GEnie, Compuserve, etc.) a BBS; or another computer, the file probably should be in ASCII format.

The reason is simple. BBSs and information services are frequented by members who own different types of computers (IBMs, Macs, Atari 8-bits, etc.) and the members may use different programs (Word Perfect on the IBM, PaperClip on the Atari XE, etc.). Only by uploading a file that's in ASCII format will you insure that it can be read and used by all users, regardless of what type of computer or program they have.

3. DESKTOP.INF FILE. If you choose to alter the DESKTOP . INF file, either from within a text editor or a word processor, you *must* save the file in ASCII format. If, instead, it's saved in some non-ASCII format, not only may the Desktop be configured improperly, but other unpleasant results may occur.

4. Sometimes, programs require ASCII files. Spelling checkers, for instance, often expect that those files consisting of lists of words be in ASCII format. In fact, it's good to adhere to

the following rule: if a support file that's associated with a program (such as a spelling checker's dictionary file, or a program's DAT file) is already in ASCII format, leave it in that form.

QUESTION: In discussing modem transfer of files, you said, "the file *probably* should be in ASCII format" before it's sent. Do you mean you can send a native format file through a modem?

ANSWER: Sure. Any file, whether it's an ASCII or native format file, or a program, is nothing more than a collection of binary digits (0s and 1s; offs and ons). The modem's job is to send 0s and 1s, and from the modem's standpoint, it doesn't matter what format the file is in.

AN ADVANTAGE OF WORD WRITER

QUESTION: I have a native format file I created in Regent Word which contains control characters that turn on the elongated print feature on my Epson printer. But when I save the file in ASCII format, the control characters are missing. I thought you said that ASCII text files could contain control characters as well as text characters.

ANSWER: ASCII files *can* contain control characters. However, some programs will not allow all the control characters to remain in a file that's saved in ASCII format. Try using another word processor, such as Word Writer.

FILE EXTENDERS AND ASCII FORMAT

Many programs are designed so that when one of their files is saved in ASCII format, a three-letter .TXT extension is automatically appended to the file name. However, not all programs follow this convention. Timework's Word Writer, for example, affixes an .ASC extender to its ASCII files.

Note, however, that the three-letter extension of a file has nothing directly to do with whether the file is in ASCII format or not. For

instance, if you create an ASCII text file in Word Writer and then change the file's name from LETTER.ASC to LETTER.RCT, the file itself won't have been changed. It is still in ASCII format and can be imported into any program that has the capability of reading ASCII text files.

TROUBLE WITH IBM ASCII FILE

QUESTION: I hired someone to do some word processing for me on her IBM computer. She saved the work as an ASCII text file, then sent it to me over a modem. The problem is that the file seems to be in some kind of weird format. When I SHOW it from the Desktop, the first line of the file gets displayed on the top line of my screen, but the second (and succeeding lines) of the file also get displayed on the top line of the screen. In other words, each succeeding line overwrites the previous line. This all happens very quickly.

ANSWER: It sounds like her word processor was the type of program that inserts only the carriage return character whenever the [Return] key is pressed. Most programs insert both the carriage return and line-feed characters. If her file had had the CR/LF combination, it would have been displayed on the Desktop in the manner that you're accustomed to. In other words, there wouldn't have been any overwriting of lines.

To confirm this diagnosis, look at the file from within a file editor (See Chapter 3). If you see only the hex value 0D at the end of lines, my diagnosis was correct. (A "normal" file will have two control codes at the end of each line: hex 0D and hex 0A. These hex values represent the carriage return and the line feed, respectively.)

You'll have to alter this file to conform to the standard that your programs are familiar with. The easiest way to do this is to use the search-and-replace function from within a word processor or text editor. What you'll want to do is replace the lone carriage returns with carriage return/line-feed combinations.

You may run into a problem, however. Some word processors and text editors won't allow you to search for carriage returns or

other control characters. You'll have to experiment with your word processor to see what its limitations are.

The program that I often use to search for or replace control codes is CONTEX.PRG, which is a shareware text editor. Control codes can be entered into Contex's search and replace dialog box by using control-key combinations. (See Appendix A to determine which Control-key combinations to use.) Figure 4.1 shows what CONTEX's search and replace dialog box looks like.

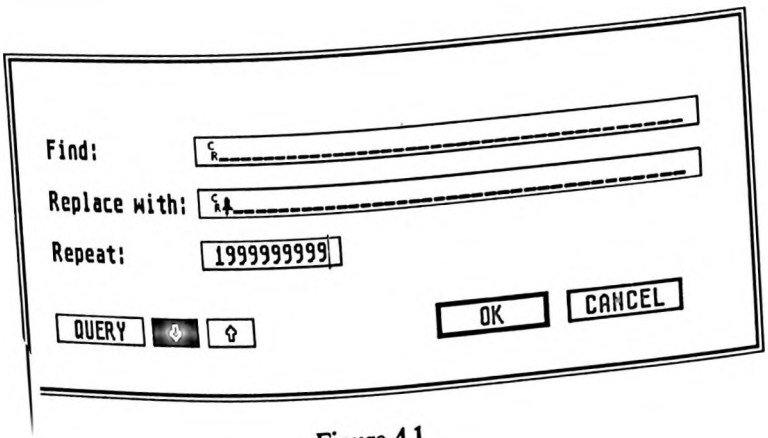


Figure 4.1

Using CONTEX, here's how you'd alter your file so it's usable on your ST:

1. First, create a backup of the file, just in case you make a mistake during the search and replace procedure. Work on the back up, not the original.

2. With the cursor on the "Find" line, press [Control]+[M]. ([Control]+[M] enters the carriage return's ASCII code. See Appendix A). The small CR symbol will appear on the screen.

3. Press [Tab] to move the cursor to the "Replace with" line. Press [Control]+[M], then press [Control]+[J]. ([Control]+[J] enters the line-feed's ASCII code.) The small CR symbol and the bell symbol will appear.

4. Press [Tab]. With the cursor on the "Repeat" line, type in a bunch of 9s to make sure the repeat procedure is performed

enough times to go through the whole file.

5. Press [Return], and when the next message box appears, press [Return] again. SAVE the file, then view it from the Desktop. It should scroll in the normal fashion.

Figure 4.2 shows how you'd perform the search-and-replace procedure from within the word processor Word Writer. It's done just about in the same manner as with CONTEX.

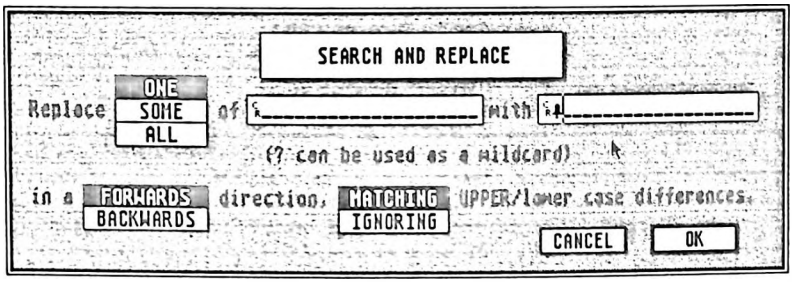


Figure 4.2.

UNWANTED BLANK SPACES

QUESTION: I downloaded a text file from an IBM bulletin board system. My ST can read the file, but it's in some strange format. Each line begins with four blank spaces. I had to go through the whole file and use the [Backspace] key to individually remove all the spaces. Is there an easier way?

ANSWER: Yes. Use the search-and-replace function from within CONTEX, or another suitable program. If you use CONTEX, here's what to do:

1. Make a backup of the file.

2. With the cursor on the "Find" line, press [Control]+[J] once, then press [Spacebar] four times. (See Figure 4.3)

3. Move the cursor to the "Replace with" line, then press [Control]+[J].

4. Insert a high enough number on the "Repeat" line, then execute the replace.

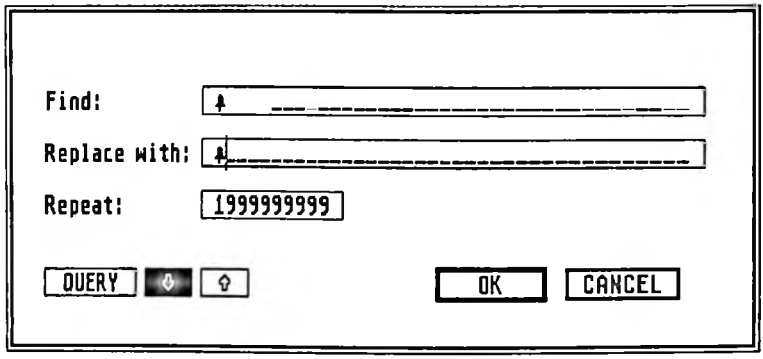


Figure 4.3

Chapter Five

DISKS and FILE RECOVERY

QUESTION: I've heard that it's possible to recover a file even after it's been dragged to the Trash can and deleted. How is this done?

ANSWER: By using a special "un-deleting program" or sector editor, it may be possible to retrieve a lost file. However, recovery will be possible only in certain situations. Often, it is impossible to go back and un-delete a file.

But before we learn the secrets of file resurrection, let's pause for awhile and cover some basics. Although it's possible to use one of the un-deleting programs without knowing anything about disk structure, it's safer to first have a basic understanding of what a disk is and how data is stored on a disk.

However, if you're impatient and are the sort of person who might say something like, "I don't want to waste time taking flying lessons. I want to solo NOW," then you can skip the following explanation. Go directly to the section further on in this chapter (page 61) entitled, "Important Warnings." For the rest of you, though, let's make sure we understand some fundamental concepts.

Our STs use 3 $^1/_2$ - inch "disks." I've put that word in quotations, since the 3 $^1/_2$ - inch by 3 $^1/_2$ - inch piece of plastic that we insert into a drive is really a *cover* for the actual floppy disk. The floppy

disk is a circular platter which rotates inside the square plastic cover.

If you'd like to see what the circular disk looks like, take an old 3 $^1/_2$ - inch disk that's been damaged, then remove the floppy disk from its cover. The cover consists of two square pieces fastened together, with the floppy disk sandwiched in between.

Using a penknife, you can pry the two halves of the square cover apart and retrieve the disk. However, don't do this with a good disk, since once you get it apart, you won't be able to reassemble it.

A circular floppy disk looks very much like a 33 $^1/_3$ RPM long playing record, except for two facts. First, a floppy disk is obviously a lot smaller, having a diameter of only 3 $^3/_8$ inches. Second, a floppy disk is made from very thin, flexible plastic. In fact, the disk is so thin that it feels almost like a piece of paper. And if you handle one, you'll notice why they're called floppies: they're very flexible.

Floppy disks are similar to long playing records in one other respect, too. On a record player, a motor turns the record, which has information stored on its surface. As the record spins, a needle, which is attached to a swinging arm, moves slowly across the record's surface and reads the recorded sounds. The needle starts at the outside edge of the record and proceeds to the inside.

In similar fashion, when you insert a disk into a disk drive, a motor in the drive spins the circular floppy disk. But instead of a needle, the drive uses one or more *heads* in order to read information from the disk. The heads travel across the disk in a manner similar to the way the arm on a record player moves across a record.

HEADS AND MAGNETIZATION

How is the information stored on the floppy disk? The disk has a very thin coating of magnetically sensitive material, and this coating is capable of accepting magnetic charges from the heads in the disk drive. The technology is similar to that which is used for music cassette tapes and video recorders (VCRs).

The heads on a disk drive record information in *digital* form. As was mentioned in Chapter 2, whenever your computer sends data to

a drive, the data is sent in binary form. This binary information consists of a stream of zeros and ones.

In the same way that an ordinary magnet has two poles -- a north pole and a south pole -- the surface of a floppy disk can be magnetized in one of two ways. When the head on your disk drive receives an "0" from the computer, the head first positions itself over a specific, small spot on the floppy disk. Then the head magnetizes that spot in one magnetic direction. If the head then receives a "1" from the computer, it magnetizes another spot on the disk in the other magnetic direction. Such magnetizations occur very quickly, since the disk is rotating at 300 revolutions per minute.

As you save more and more files to a disk, thousands and thousands of "north magnetizations" and "south magnetizations" get stored on the disk's surface. And just as you can take a cassette music tape and erase an existing song from it, or record a new song over an old song, you can delete files from a disk, or cause one file to be written over an existing file. Deleting or writing over a file merely involves causing the head to change the pattern of 0s and 1s in specific spots on the disk.

READING AND WRITING

The heads in a disk drive are capable of performing two types of activity: *writing* and *reading*. Writing to a disk is analogous to recording on a cassette tape player, and reading is analogous to playing.

If you're using a word processor and you want to save what you've typed, your work gets *written* to the disk as a file. Later, when you want to load that file into your word processor again, the disk's heads *read* the file from the floppy disk and the file's contents get displayed on the screen.

But there's a significant difference between a cassette music tape and a floppy disk. The music on the cassette tape is usually listened to during one session, from beginning to end. This is true even though there may be eight or ten separate songs on the tape. In this sense, a cassette tape could be considered a *sequential* storage

medium, since it is recorded on and listened to from beginning to end.

Contrasted with this, our computer disks are *random access* storage devices. Because of the way disk technology is designed, we're able to immediately access any file on a disk, regardless of whether the file is physically stored on the outer edge of the disk, somewhere near the middle of the disk, or wherever.

The heads in a disk drive can easily position themselves over any spot on the disk, then read data from (or write data to) the disk, then quickly position themselves over another spot on the disk and read or write data.

FORMATTING

But in order to be able to do this, the surface of the disk has to be formatted. Formatting is a kind of mapping, or breaking into segments. To understand formatting, let's consider an analogy. Let's think about what we would do if we wanted to create a new town out in the middle of some undeveloped land.

Our first step would be to survey the area and break it down into segments. Streets would be created, and each street would be assigned a name. We might call one street *First Street*, the next one *Second Street*, and so on.

After the streets were laid out, we'd break each street down into even smaller segments, which would be individual plots of land. Each plot would have a house and would be assigned a number, with the first house on the street probably designated as house number *1*, the second house as house number *2*, and so on.

Formatting a disk is similar to this. First, the disk's surface is broken down into *tracks*. Tracks are concentric rings on the disk's surface. (See Figure 5.1) To help remember what a track is, I think of a circular racetrack where runners are positioned in a particular track, or concentric circle.

On a typical ST floppy disk, 80 tracks are created. Each track is assigned a number, just like streets are assigned names. The 80 tracks are numbered from 0 to 79, with track 0 near the outside edge of the disk, and track 79 nearer the center. If you could see

the tracks on a disk (which you can't), the disk would look sort of like a target with a bull's eye.

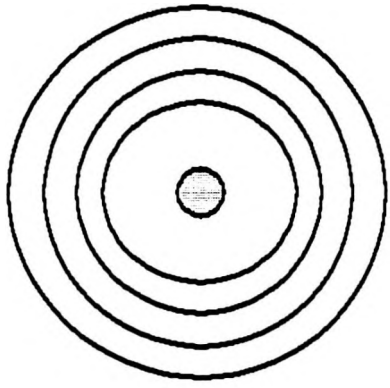


Figure 5.1

Formatting also involves breaking down each track into *sectors.* Each sector contains exactly 512 bytes, which amounts to 512 characters. On a typical ST floppy disk, each track has nine sectors. (See Figure 5.2) To help remember the difference between sectors and tracks, I think of sectors as being similar to sections of a pizza pie that has been cut into nine pieces.

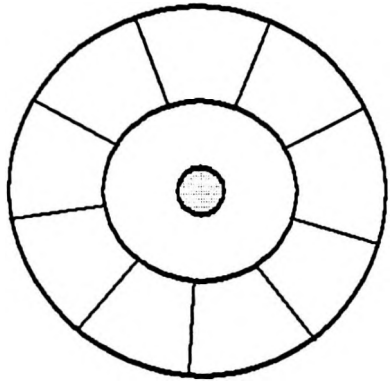


Figure 5.2

There are nine sectors on one side of a disk, and each sector is assigned a number from 1 to 9. Therefore, a single-sided disk that's formatted from the Desktop will have nine sectors. A double-sided disk will have 18 sectors: sectors 1 through 9 on the first side and sectors 1 through 9 on the second. (If a disk is formatted with a high density formatting program, more sectors may be created: 1 to 10, or 1 to 11, etc.)

The fact that all the tracks and sectors have been assigned a number makes it easy to refer to a particular spot on the disk. In the same way that someone could say, "Meet me at 2nd Street, house number 124," the computer can tell the disk drive, "Retrieve the file that's located at track 7, sector 3."

CLUSTERS: THE SMALLEST STORAGE UNITS

For various reasons, sectors are often dealt with in groups of two. On a standard floppy disk, two adjacent sectors constitute a *cluster.* Just as tracks and sectors are numbered, clusters are assigned numbers from 2 on up, until the end of the disk is reached. Since a sector can hold 512 bytes, and since there are two sectors in a cluster, each cluster can hold 1024 bytes (i.e., characters).

This number is important. The fact that a cluster is 1024 bytes large is significant because a cluster is the smallest storage unit on a disk. In other words, no matter how small your files are, a cluster will be able to hold only one of your files.

For instance, if you have two files, each of which is 360 bytes long, one of the files will be stored in a particular cluster, and the other file will be stored in another cluster. Each file will be put in its own cluster, even though a single cluster (which is 1024 bytes in size) has room for the two 360 byte files.

Normally, of course, you never need to know anything about tracks, sectors, clusters, or disk formats. The ST's operating system allows you to perform the basic functions -- saving, copying, deleting or reading a file -- without having to know much more than how to move and click the mouse. But as soon as you want to go beyond the basics of Desktop file functions, a knowledge of tracks and sectors becomes useful.

As you know, when you double-click on a drive's icon from the Desktop, the names of all the files on the disk are listed in the window. Actually, though, there's more on a disk than just the listed files.

A number of sectors at the beginning of a disk contain some very important "non-file" material. The first sector on a disk -- regardless of whether it's a floppy disk or a hard disk -- is the boot sector. Since you don't need to know anything about the boot sector in order to recover a deleted file, we won't be discussing it right now. (The boot sector is explained briefly at the end of this chapter.)

THE DIRECTORY AND THE FAT

Next to the boot sector are a number of sectors that are reserved for two kinds of "indexes" of what's on the disk. One of the indexes is called the disk *directory*; the other is the *File Allocation Table*. The File Allocation Table is commonly referred to as the "FAT."

The directory contains the same information that appears in a disk's window: the names of each file on the disk, the date and time of creation, and the file size. The directory also contains some "file attribute" information, such as whether the read-only flag has been set for the file, whether the entry is a data file, a folder, or a disk name.

Most importantly, the directory contains the address of the cluster where the beginning of the file is stored. This address is called the *starting cluster number*. The FAT, on the other hand, contains a list of all the clusters on the disk, and what those clusters are being used for.

WHEN FILES GET BROKEN UP

Knowing where the beginning of a file is located is necessary in order for the computer to be able to instruct the disk head where to position itself when it wants to read a file. But knowing where a file

starts is often not enough. Sometimes, a file may have been broken up into many small segments which are scattered in different places on the disk.

Why would a file be broken up? To understand this, let's examine what occurs when a file is saved to a disk. Let's assume that you've just formatted a new disk, and you're now writing a short document in your word processor that contains 360 characters. When you use the mouse to choose SAVE from the word processor's pull-down menu, here's what happens:

First, the disk head positions itself over the first unused sector on the disk which is capable of receiving a file. The disk spins; then the head starts writing the file to the disk, sector by sector. Since your document is only 360 characters long, it will take up only one sector on the disk. (A sector, you'll remember, can store 512 bytes, i.e., 512 characters.)

Since you're saving the file to a newly formatted disk on which there are no other files, the first available sector is sector 1 on track 2. (Tracks 0 and 1 contain the boot sector, two copies of the FAT, and the directory.)

Let's assume that you now compose two other 360 byte documents, each of which you also save to disk. The second document will be saved to sector 3 on track 2, and the third document to sector 5 on track 2.

Let's digress for a moment. One might ask, since the 1st document (which took up only one sector) was saved to sector 1, why wasn't the second document saved to sector 2?

EMPTY BUT UNAVAILABLE SECTORS

The answer is, the basic storage unit on a disk is the cluster, which consists of two sectors. The cluster of which sectors 1 and 2 are part was not "available," even though its second sector was completely empty. It was considered unavailable, since its first sector already had a file in it. The first available cluster of which *neither* sector has anything in it, is the cluster consisting of sectors 3 and 4. This explanation also covers why the third file was saved in sector 5.

TRASHED FILES ARE NOT DELETED

Continuing with our hypothetical example, let's assume that, after printing out document number 2, you decide you don't need that file any more, so you delete it by dragging its icon to the Trash can. Contrary to what many people think, dragging a file to the Trash can does *not* delete the file. All that happens is that the entry in the directory, as well as the cluster list in the FAT, are changed. But the characters that constitute the file remain unchanged.

In other words, even though you've trashed file number 2, nothing is changed in sector 3 of track 2. It's as if the librarian at your public library, deciding to remove a book from circulation, deletes the volume's 3x5 index card from the card catalog, but leaves the book on the shelf. Although the file is still in track 2, sector 3, the entry in the FAT says that this sector is now unused, and is thus available for storage.

Let's assume that you now open your word processor and write a fourth document. But this time, you write a relatively large document, one that has about 1,500 characters in it. It's thus much bigger than each of your previous documents.

When you save this file, here's what happens. The computer looks for the first available cluster on the disk. That would be the cluster containing sectors 3 and 4 on track 2, which was made available when you trashed file number 2. The first 512 bytes (i.e., characters) of this fourth document get saved to sector 3, and the next 512 bytes get saved to sector 4. But since the document was 1,500 characters long, there's still data remaining to be saved.

Consequently, the computer looks for the next available cluster on the disk. The FAT tells the computer that the next cluster (which contains sectors 5 and 6 on track 2) is *not* available (since it's holding document number 3) but that the next cluster (which contains sectors 7 and 8) is unused. Therefore, the remaining portion of your latest document gets sent to track 2, sector 7. (See Figure 5.3.) In other words, the first part of the file is in sectors 3 and 4, and the rest of the file is in sector 7.

There's a name for what's happened to this disk: *file fragmentation.* A fragment of the file resides in one sector, and another fragment resides in another, *non-contiguous sector.*

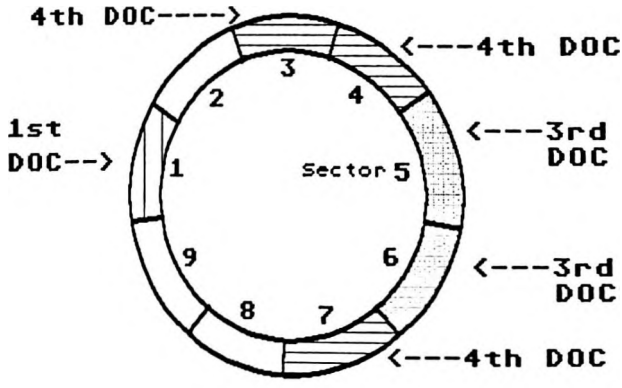


Figure 5.3

This example of fragmentation, however, is a very mild one. It's common for files to be fragmented into many different sectors throughout a disk. The more deletions and saves that have been performed on a disk, the greater degree of fragmentation there is likely to be. If you've used a disk a lot, you have probably deleted files, saved files, re-deleted files, and re-saved files so many times that there may be very few data files which are saved in blocks of contiguous sectors.

This sort of situation points out the importance of the FAT. The file allocation table consists of a list of all the clusters on a disk, with a notation for each cluster as to whether it's being used or is free.

The FAT also contains a notation for each cluster which indicates the next cluster of the file. In other words, the FAT is used to link together all the clusters that comprise a file. To understand this, let's see what happens when we tell the computer to load a file from disk into memory.

First, the operating system looks at the directory to find out where the first cluster of the file is located. Let's say that the directory indicates that the file starts at cluster 5. When the

operating system looks in the FAT's entry for cluster 5, it learns what the next cluster is. Let's assume this next cluster is number 7.

When the operating system looks in the FAT's entry for cluster 7, it learns where to find the next cluster. In this way, the FAT entries for each cluster guide the operating system to each succeeding cluster until a cluster is reached which contains an "end of file" entry.

The file allocation table is so important that two copies of it are kept on each disk. If the first copy is damaged, the second FAT can be used to locate and recover the files on the disk.

OK, enough theory. We now have enough knowledge about disk structure to recover a deleted file. However, before proceeding, here are some important suggestions.

IMPORTANT WARNINGS

1. If you discover that you've mistakenly deleted a file, immediately STOP WHAT YOU'RE DOING. It's vitally important that you do NOT save (or update) a file on that disk. When you deleted the file, you caused the FAT entries for that file's clusters to be changed to "free" (i.e., unused). If you now save a file to the disk, it's possible that the saved file will be sent to some or all or your deleted file's clusters. If this happens, your deleted file really will be deleted, permanently. When a new file is written over an old one, the old file is lost forever.

2. If the file that you mistakenly deleted was in a RAM disk, LEAVE THE COMPUTER ON. If you turn off the computer (or press [Reset]) the file will be lost forever.

3. If your deleted file was on a floppy disk, make a copy of that disk. Then use your un-deleting program or sector editor only on the copy. Trying to recover a deleted file can be tricky, and the actual recovery attempt can itself destroy a file, or a disk. To make a copy of the disk, use the mouse to drag the icon of the original disk on top of the icon of a newly formatted disk.

4. Before you try to recover an important lost file, practice. Save some files to a disk, then delete one or two files. Then use

your un-deleting program or sector editor and try to retrieve the files. It's better to gain experience on a test file than on an irreplaceable one.

Enough warnings. Let's proceed.

I'm going to explain how to use two different programs to recover a deleted file. The first program, MichTron's *Utilities*, is a commercial product, and it provides the potential of recovering even a very long file.

But since you'll have to buy MichTron's *Utilities* in order to use it, I'll also be describing how to use the public domain program, TINY TOOL, to recover very short files.

THE MICHTRON *UTILITIES* PROGRAM

One advantage of the MichTron program is that it is very easy to use. In fact, even without reading any part of this chapter up to this point, you'd be able to use the following instructions to recover a file.

A second advantage of the program is its versatility. It can retrieve files from single sided or double sided floppies, RAM disks, or hard disks. And finally, it works on both monochrome and color systems.

Let's assume that you have mistakenly deleted the 4th.DOC file. Here's how to use the MichTron program to retrieve it:

1. Make a copy of the disk containing the deleted file. Double-click on the MichTron program.

2. Choose "File Attributes" from the ATTRIBUTES Menu-bar.

3. Choose the letter that corresponds to the drive containing your disk.

4. A window will appear showing existing files on the disk, as well as any deleted files that remain. (See Figure 5.4.)

Note that the names for the 1ST.DOC and 3RD.DOC files are complete, since these are files that have not been deleted. If a file *has* been deleted, the first character of the filename will appear (in the MichTron program) as a question mark. As you

can see, our 4TH.DOC file appears as "?TH.DOC", indicating that it is a deleted file.

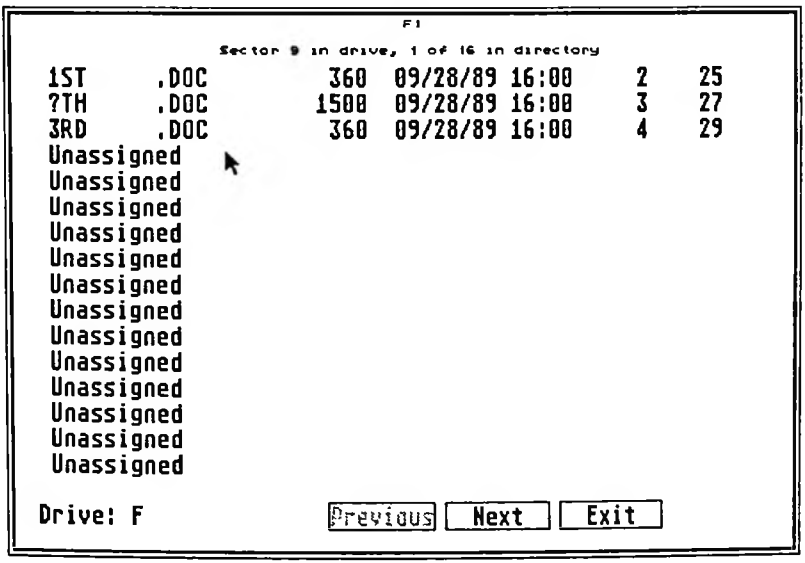


Figure 5.4

5. To resurrect the 4TH.DOC file, mouse-click once on that file's filename. Another window will appear (See Figure 5.5). This window shows part of the contents of the cluster that contains the beginning of the file.

6. To retrieve this cluster, mouse-click on the USE button that's located in the lower right-hand corner of the window.

7. Another window will appear, showing the cluster that contains the next segment of the file. As before, mouse-click on the USE button.

8. At this point, you'll notice that a sentence at the top of the window states, "You have recovered 2 clusters of the required 2." (The number of clusters you'll need to recover will, of course, vary, depending on how long your own file is.) Since we've recovered all of clusters in the file, we can mouse-click

on the QUIT button.

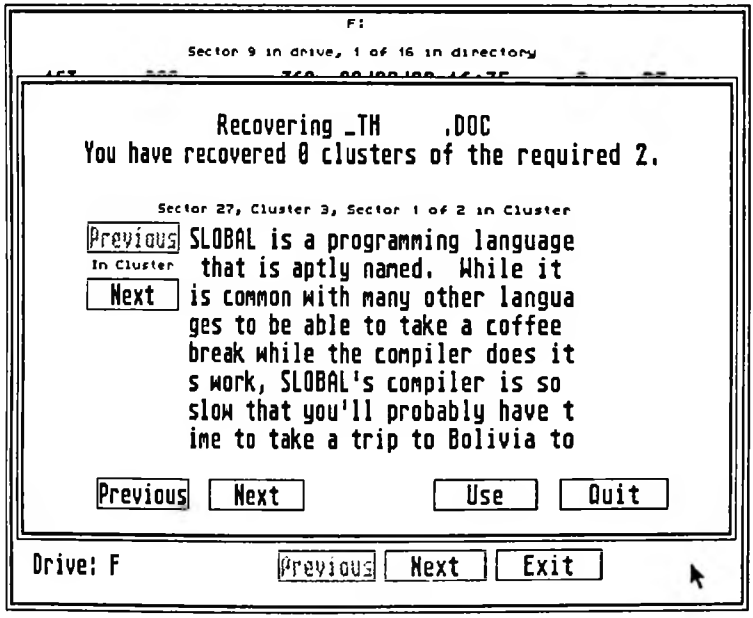

F:
Sector 9 in drive, 1 of 16 in directory
Recovering _TH .DOC
You have recovered 0 clusters of the required 2.
Sector 27, Cluster 3, Sector 1 of 2 in Cluster
Previous
In Cluster
Next
SLOBAL is a programming language that is aptly named. While it is common with many other langua ges to be able to take a coffee break while the compiler does it s work, SLOBAL's compiler is so slow that you'll probably have t ime to take a trip to Bolivia to
Previous
Next
Use
Quit
Drive: F
Previous
Next
Exit

Figure 5.5

9. Another window will appear, giving us the choice to CANCEL, RESUME, or SAVE. Mouse-click on the SAVE button.

10. Still another window will appear, asking if we want to update the disk. Mouse-click on YES.

11. Our file has now been recovered, and we're returned to the initial *Utilities* window. (See Figure 5.6.) The question mark, which had been at the beginning of the file's filename, has been replaced by the underline character. If you want to change the name of the file back to its original name, you can do so when you return to the Desktop. (To do so, click once on the file's name, choose SHOW INFO from the FILE Menu Bar, then make the alteration.)

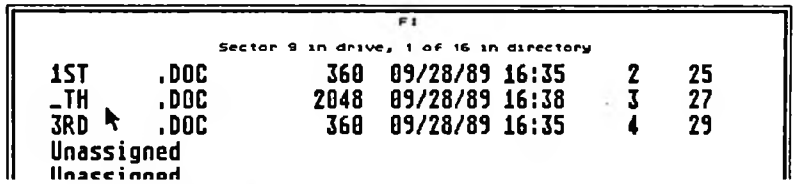

F1
Sector 9 in drive, 1 of 16 in directory
1ST .DOC 360 09/28/89 16:35 2 25
_TH .DOC 2048 09/28/89 16:38 3 27
3RD .DOC 360 09/28/89 16:35 4 29
Unassigned
Unassigned

Figure 5.6

In the process of recovering the file, a few extra bytes have been added to the end of the file, bringing its size up to 2048. We can remove these extra bytes by performing a CUT operation from within a word processor.

USING A SECTOR EDITOR

Whereas the MichTron program is so easy to use that someone with no knowledge of disk structure can use it to recover a file, a sector editing program requires a fair amount of knowledge. As a minimum, you need to be able to enter in track and sector addresses, as well as be able to change a byte by entering a hexadecimal or decimal value. However, by using the following instructions, many of you should be able to successfully recover a file, provided that it's a short file.

A number of free public domain sector editing programs exist, but since we've already discussed the file editing features of the TINY TOOL program, let's spend a little time with the sector editing capabilities of this program.

Although you can use a sector editor to retrieve both small and large files, I'm going to be explaining only how to recover a small (less than 512 byte) file. The recovery of such a file, which occupies only one sector, can be accomplished by changing just one byte in the disk's directory. Although you can use a sector editor to recover large files, the changes that are required in the FAT in order to do so are fairly delicate, and are beyond the scope of this chapter.

Before we run the TINY TOOL program, we need to know where on a disk to look for the directory. (Although for the simple

exercise that follows, we don't need to know where the File Allocation Table, or Boot sector, is located, those addresses are nevertheless contained in the following information.)

Where the directory is located on a disk will depend on what type of disk we have, and what type of formatting program was used. If we have a single-sided disk that was formatted from the Desktop using TOS's standard formatting routine, here's where things are located:

```
Track 0, Sector 1          = Boot sector
Track 0, Sectors 2-6       = FAT (copy # 1)
Track 0, sector 7 to
    track 1, sector 2      = FAT (copy # 2)
Track 1, sector 3 to
    track 1, sector 9      = Directory
Track 2, sector 1 to
    track 79, sector 9     = Files
```

If we've formatted a double-sided disk from the Desktop, here's where things are located:

```
Side 0, track 0, sector 1          = Boot sector
Side 0, track 0, sectors 2-6       = FAT (copy #1)
Side 0, track 0, sectors 7-9  and
    side 1, track 0, sectors 1-2   = FAT (copy #2)
Side 1, track 0, sectors 3-9       = Directory
Side 0 & 1, track 1, sector 1 to
    track 79, sector 9             = Files
```

I've referred to the sides of a double-sided disk as 0 and 1, since that's the form in which you tell the TINY TOOL program which side of the disk to look on. However, many people refer to the sides of a disk as A or B.

Now we'll use TINY TOOL to recover a file. Let's assume that I saved three 360 byte files to a single-sided disk. One of the files, which I had named "2ND.DOC," I then deleted. To recover the deleted file, here's what to do:

1. Make sure that a copy of the TINY TOOL program, which is a desk accessory, is on your boot disk, then turn on the computer with the boot disk in drive A.

2. Put the disk from which the file was deleted into one of your disk drives. (I'm going to put it in my external drive.)

3. Use the mouse to choose TINY TOOL from the DESK Menu bar. When the initial TINY TOOL window appears (See Figure 3.1 in Chapter 3) mouse-click on the sector button.

4. A second TINY TOOL window will appear which contains a white rectangular box. The word "Drive:" appears in the left side of the box, and this is a sign that TINY TOOL wants to know in which drive my disk is located.

I could respond with one of two answers: 0 or 1. If I type 0, I would be indicating that I wanted to look at the A: drive. But since I've put the target disk in my external drive, I type 1, then press [Return].

5. "Drive: 1" gets displayed in TINY TOOL's window, and the word "Side:" appears. TINY TOOL is asking me what side of the disk I want to examine.

I can respond with one of two answers: 0 or 1. If I type 0, I'm indicating that I want to examine the first side of the disk. I'd type 1 if I wanted to look at the second side. Since my single-sided disk has only one side, I press 0, then press [Return].

6. "Side: 0" gets displayed in TINY TOOL's window, and the word "Sector:" appears. The program wants to know what sector to read. Since I want to examine the directory, I type 3, then press [Return].

7. "Sector: 3" is displayed, and the word "Track:" appears. Since the directory on a single-sided disk is located on sector 3, track 1, I type 1, then press [Return].

8. The red light in my external drive comes on and the contents of sector 3, track 1 are read into the computer's memory. This information is also displayed in the TINY TOOL window. (See

Figure 5.7)

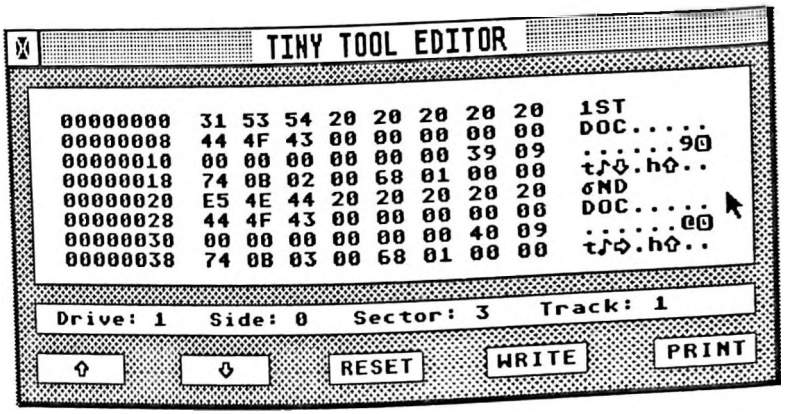


Figure 5.7

9. Only the first 64 bytes of the sector are displayed in the window, but you can mouse-click on the down arrow to view the rest of the sector's contents.

On the right-hand side of the window is an ASCII display of the sector's contents. The middle group of characters displays the hexadecimal equivalents of the ASCII values.

As you can see, the top line of the ASCII display reads "1ST," and the second line reads "DOC." This is a display of the name and extension of the first file that was saved to the disk.

The first character in the fifth line of the ASCII display contains a strange character that looks somewhat like an o that has a feather sticking out of it. This character is decimal 229, which is the same thing as hex E5. If all of this sounds pretty weird, go back to Chapters 2 and 3 and re-read the sections on the hexadecimal numbering system and file editors.

The presence of the hex E5 character indicates that this file has been deleted. In other words, whenever a file is deleted, the first character of the file's name is changed (in the directory) to hex E5. If we use TINY TOOL to change this hex E5 to a letter of the alphabet, or to a number, we will recover the first

sector of this file. Here's how to make the change:

10. Mouse-click on the E5, which are the first two characters of the fifth line in the TINY TOOL window's middle group.

11. A box and prompt will appear which says, "Actual value: E5....New value:" (See Figure 5.8) This prompt is asking you what value you would like to replace hex E5 with. Although we can insert any letter or number we want, why don't we insert the number "2"? Doing so will change the name of the file back to the way it was before it was deleted; in other words, back to "2NDDOC."

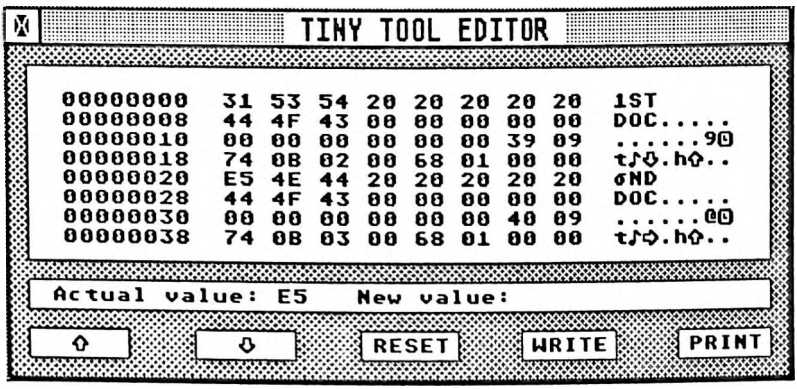


Figure 5.8

There are two ways that we can input an ASCII "2" into TINY TOOL. We can type the hex value of ASCII 2, or the decimal value of ASCII 2. Let's use the decimal value.

By referring to Appendix B, you'll find that ASCII 2 is the same as decimal 50. Therefore, press 5, then 0, then [Return].

The display of the contents of the sector will be refreshed, and you'll notice that the ASCII "o with the feather in it" has been replaced with a 2.

12. Now mouse-click on WRITE, which will write the change to the disk.

Note that if you had wanted to input an ASCII 2 by entering a hex value, you would have typed the dollar sign [$] followed by the number 3 and the number 2. Using the dollar sign tells TINY TOOL that you're going to enter a hex value, and 32 is the hex value of ASCII 2.

13. You can now close the TINY TOOL window by mouse-clicking on the Close Box, which is in the upper-left hand corner. Then return to the Desktop and call up a window for the disk. If you've done everything correctly, the previously deleted file should now be listed.

OTHER RECOVERY PROGRAMS

There are other programs, both commercial and public domain, with which you can recover a deleted file. There's also another public domain file and sector editing program called MemFile (MEMFIL20.ACC and MEMFIL20.RSC). (See Figure 3.6 in Chapter 3)

NOTES

The boot sector contains information about how the disk is configured, such as whether it's single or double-sided, whether it uses a standard or non-standard format, what the serial number of the disk is, how many tracks and heads are on the disk, how many sectors on each track, how many sectors in each cluster, and so on. The boot sector can also contain an instruction that causes a program to run automatically.

For instance, the boot sector can cause the operating system to be automatically loaded into the computer's memory. When the Atari ST first came out, its operating system had not yet been put in a ROM chip. Therefore, in order to use the computer, you had to start up your ST with a special boot disk in drive A which contained the operating system.

The first sector on the boot disk included an instruction that caused the operating system to be loaded into the computer's

memory. This procedure is called *booting*, since the computer is "pulling itself up by its boot straps." (The computer is loading the very program that is necessary for it to operate.) Now, however, since the operating system is in ROM, our disks no longer contain the boot program.

Chapter Six

HARD DRIVES

There were three reasons why I resisted buying my first hard drive. First, I'm stingy. I didn't like the idea of spending hundreds of dollars on an accessory, especially when the original price of my whole ST system had been so low.

Second, I was afraid of hard drives. I had worked in an office that had four hard drives hooked up to Macintosh computers, and it seemed that every two months or so, one of the drives was being sent back for repairs.

Third, I wasn't convinced that I needed a hard drive. My ST's wo floppy drives were serving me well, and I was still awed by the speed of a RAM disk.

Although I kept hearing comments like, "Once you get a hard drive, you'll wonder how you ever got along without one," I wasn't convinced. Just because I'd get addicted to a hard drive didn't necessarily mean that buying one would be a smart move.

But I finally succumbed, and I now find myself relying more and more on my hard drive. Starting with the basics, let me share with you some of the ins and outs of hard drive management that I've learned. Then, in the next chapter, we'll cover some advanced subjects that have to do with addressing multiple drives and assembling a hard drive system from individual parts.

THE BASICS

Hard drives are like floppy disks and RAM disks: they're devices

for storing data. The main purpose of a hard drive or a floppy disk is to store programs and data files in a permanent way that will allow you to shut off your computer.

Hard drives are sometimes called hard disks, Winchester drives, or "fixed disks." This last term points out one of the main differences between hard disks and floppy disks. With a standard floppy disk drive, you can pop one floppy out and insert another. With a hard drive, though, the actual disk remains fixed. In fact, a hard drive's disk and heads are sealed inside a chamber that is never opened by the user. By permanently enclosing the disks, dust and other foreign matter are excluded.

Whereas a floppy disk is made of thin flexible plastic, a hard drive's disks are made of metal (usually about $^1/_{16}$-inch thick).

Like floppy drives, a hard drive's disk has a thin coating of material that is magnetically sensitive, as well as heads that can write and read data from the disk. But whereas a floppy drive's heads actually touch the surface of the disk, a hard drive's heads float on a cushion of air that's thinner than a human hair.

ADVANTAGES OF HARD DRIVES

Users like hard drives because they're fast, they hold a lot of data, and they're reliable.

Whereas floppy drives spin at from 288 to 300 rpm (revolutions per minute), hard drives spin at about 3600 rpm. Additionally, the disks in a hard drive spin continuously, whereas the disks in a floppy drive only spin when the computer accesses them.

In addition to rpm, there are other factors which determine how quickly a hard drive will process data. For instance, many hard disks have more than one platter (i.e., disk), and there's usually one or two heads on each side of a platter. More platters and heads mean that the drive will take less time to access data.

Later on in the chapter I'll discuss a number of additional factors which contribute to just how fast a hard drive is.

CAPACITY

Hard drives are usually referred to by the amount of data that they store. The measuring standard is megabytes, so you may hear a user say, "I bought a 20-meg hard drive," or, "He's moving up to a 60-meg drive."

How much data will a hard drive hold? A 20 megabyte drive can store the contents of about 28 double-sided disks (or 56 single-sided disks). You'll be able to put many or all of your programs and data files on your hard disk.

In my case, I found that after I bought a hard drive, my computer desk was a lot less cluttered. I no longer had to, for instance, hunt for the word processing disk, then look for the appropriate data disk, then hunt for the telecommunications disk, etc.

RELIABILITY

As mentioned, my experience with a particular brand of hard drives for the Macintosh led me to believe that fixed drives were unstable. However, according to the computer authorities that I've consulted, hard drives are actually *more* reliable than floppy drives. My own experience corroborates this. Before I acquired my hard drive, my screen occasionally displayed the frightening message, "Data in drive B may be damaged." However, I have yet to lose a file on my hard drive as the result of hardware failure. (I have, however, lost a whole hard disk full of files because of human error.)

Of course, since you store a lot more files on a hard disk than on a floppy, if your hard disk *does* crash, you'll lose a lot more data. It's sort of like flying. Although a jumbo jet is a lot safer than a two-passenger plane, if the jumbo jet goes down, the loss is greater.

But if you use your drive properly, and backup important files regularly, you'll probably find that a hard drive is a very reliable medium.

HARD DRIVE COMPONENTS

A hard drive system for your ST will consist of three main components: host adapter, controller, and hard drive mechanism. (See Figure 6.1) There also will be cables, software, a power supply, and possibly a separate case to house the components.

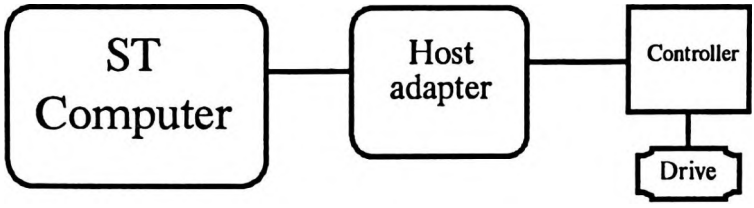


Figure 6.1

If you've purchased an already-assembled hard drive system (or if your ST, itself, contains a hard drive) you may never see the individual components that make up the hard drive system. But it will still be good for you to know a little about the individual parts that make up your hard drive system.

The hard drive mechanism consists of a sealed case containing the magnetically sensitive platters on which the data is stored.

The controller is a printed circuit board which regulates the hard drive unit. The controller receives write, read, and other commands from the computer, interprets these commands, then sends appropriate instructions to the hard drive unit.

The host adapter is a separate printed circuit board which converts the ST's ACSI signals into the SCSI standard that most drive units and controllers can understand. (ACSI stands for Atari Computer Systems Interface, and SCSI, which is pronounced "scuzzy," stands for Small Computer Systems Interface.)

Often, the controller and the hard drive mechanism are separate units. You might buy the controller from one source, then order the hard drive from another source. You could also swap a drive (or a controller) from one system to another.

There are also what are called "hard drives with embedded SCSI

controllers." These are integrated units that contain both the hard drive mechanism and the controller. One of the advantages of this type of unit is price. Buying a drive with embedded controller is often cheaper than purchasing a controller and drive separately.

HOW BIG A DRIVE TO BUY

Once you've made up your mind to buy a hard drive, you'll have to decide whether to purchase an already assembled system, or to put together your own unit. The easiest and safest alternative is to buy an assembled system that has been made specifically for the ST. I decided, however, to assemble my own unit, and have described how I did it in Chapter 7.

Whichever option you choose, your next step will be to decide how big a drive to get. Although it's possible to hook up something as small as a 5-meg drive to an ST, you probably shouldn't bother with anything less than a 20-meg drive. Since one of the major reasons for using a hard drive is the increased storage capacity, it doesn't make much sense to spend the money for a drive system that can hold only six or seven floppies worth of data.

Also, since the cost of the hard drive mechanism (i.e., the part that is 5-megs, or 20-megs, or whatever) is only part of the total cost of the whole hard drive system, even if you decide to save money by buying a small capacity drive mechanism, you'll still be faced with the fixed costs of controller, host adapter, etc.

Another reason for buying a large hard drive mechanism is that the price *per megabyte* for a larger drive is often lower than for a smaller drive. For example, the price of a 60-meg hard drive mechanism may be considerably less than twice the price of a 30-meg unit.

However, not everyone agrees that you should buy the largest hard drive you can afford. Some users, for instance, would prefer to have two 20-meg drives rather than one 40-meg drive. They feel safer with two independent drives, since it's less likely that both drives would suffer data loss at the same time. And if one drive *did* go bad, there would still be one usable drive while the other was being repaired.

SETTING UP YOUR DRIVE

The physical hook-up of a hard drive is easy. If you've purchased a ready-made system, just plug its cable into the DMA (Direct Memory Access) port on the back of the ST. (The DMA port is labeled "Hard Disk" on the back of the ST.) If you've assembled your own system, plug the interface cable from the host adapter into the DMA port.

But even after your hard drive is connected, chances are you'll still have to perform one or more of the following steps before you can use your new drive:

- format the drive.
- partition the drive.
- zero out the directories (if necessary).
- run the hard drive booting program.
- create Desktop icons for the partitions.
- save the Desktop.

FORMATTING

Some of the drives made specifically for the ST are shipped from the factory already formatted and partitioned. If you have one of these systems, you may be able to start using the unit as soon as you plug it in. However, many drives do not come formatted, and just like floppy disks, their surfaces will need to be mapped out into tracks and sectors before they can be used.

Each of the manufacturers of hard drive systems for the ST (Atari Corp., Supra, ICD, Berkeley Microsystems, Future Systems, etc.) includes formatting and partitioning software with their units. Although it may be possible to take the software that's designed for one system and use it on another system, it's probably safer not to do this. (In some cases this will *not* be possible.) And although the different manufacturers' programs are fairly similar, you should follow the specific formatting instructions that came with your hard drive or host adapter.

PARTITIONING

At some point during the formatting procedure, you'll have to decide how you want to partition your drive. To partition a hard disk means to divide it into two or more separate storage areas. For instance, if you have a 20-meg drive, you might decide to create four partitions, each of which could be five megabytes large.

The decision of how many partitions to create, and how large each partition should be, is generally up to you. If you wanted to, you could divide that 20-meg drive into six partitions of the following sizes: 4, 4, 6, 3, and 3-megs.

However, you don't have total freedom in choosing partitions. The following constraints will limit your options:

1. With the older versions of TOS, a hard disk directory is incapable of mapping out more than 16 megabytes worth of files. This means that if your hard drive is larger than 16 megs, you'll *have* to partition it (assuming you want access to all the sectors on the drive). (With TOS 1.4 or higher, you can allocate up to 32 megabytes per partition.)

   For example, if you have a 20-meg drive, you'll have to create at least two partitions. A 40-meg drive will require at least three partitions. If, however, you've acquired one of the older 10-meg drives, you won't need to divide it into separate partitions. A 10-meg drive can consist of one partition of 10 megabytes. (Of course, you *could* divide it into multiple partitions.)

2. At present, you can not have more than 14 hard disk partitions, nor can any single hard drive have more than 12 partitions. However, these restrictions are merely conventions, not limitations of TOS.

Each partition on a hard drive is treated by your computer as if it were an independent drive. A drive letter is assigned to each partition, in the same way that your internal floppy drive is assigned the letter A, and a second floppy is assigned the letter B.

Consequently, if you have, for instance, four partitions, your Desktop will contain six disk icons: one each for floppy drives A and B, and icons for hard disk partitions C, D, E, and F. Your

computer will think you have two floppy drives and four hard drives.

One warning note: if you decide to re-partition your drive, first make sure that any important files on the drive have been backed up, since re-partitioning will cause all the data on the drive to be lost. Not even a file recovery program will be able to bring back files that were on a disk that has been re-partitioned or reformatted.

PHYSICAL vs LOGICAL DRIVES

Two special terms are sometimes used to differentiate partitions from drives. The hard drive mechanism that contains platters and heads is called the *physical* drive, and the electronic divisions (i.e., the partitions) are called *logical* drives.

For instance, if a 40-meg drive is partitioned into four 10-meg drives, there are four logical drives, which together form the physical drive. If, however, you had two 10-meg drives, labeled "C" and "D", each drive designator (i.e., each partition) would represent a physical and logical drive at the same time.

ZEROING OUT PARTITIONS

It's conceivable that, even after you have formatted a hard drive, there may be some garbage in a directory. If so, you'll want to zero out that directory.

Zeroing out is the act of erasing a specific partition on your hard drive. Unlike formatting, which results in erasure of the data contained on *all* of a drive's partitions, the erasure that zeroing out performs is confined to a specific partition.

Zeroing out is brought about by choosing the appropriate dialog box button from within one of the hard disk utilities programs (Supra's SUPUTL.PRG, ICD's HDUTIL.PRG, etc.).

You can zero out a partition any time you want, not just after you've formatted your disk. For instance, if you have a partition full of files that you no longer want, it may be a lot quicker to zero

out the partition than to drag all the files to the Trash Can.

RUNNING THE HARD DISK BOOTING PROGRAM

Even after formatting and partitioning your hard drive, your ST still won't be able to communicate with the drive unless the hard disk has been "booted."

Booting a hard disk simply means running a special program called a hard disk booter, hard drive handler, or hard disk driver. The handler tells the computer that a hard disk is present, and allows the computer and disk to communicate with each other.

If, when you try to access a hard drive by double-clicking on one of its partition's icons, you get a message saying, "The drive you specified does not exist," it probably means that you haven't run the hard drive booting software.

There are a number of different booting programs, and which program you use should normally be determined by the kind of host adapter your system contains. For instance, ICD owners use the ICDBOOT.PRG handler; SupraDrive owners use SUPUTL.PRG; Atari owners use AHDI.PRG, etc.

THREE WAYS TO RUN THE HANDLER

There's more than one way to run a hard drive handler. The three ways discussed below are listed in their order of complexity, with the simplest method listed first.

1. Running the program "manually" from a floppy drive. To do this, insert into your A or B drive a floppy disk containing your hard drive handler program. (The program can be either in the root directory or in a folder.) Then double-click with the mouse on the handler's name or icon.

   After the handler has been run, try accessing the hard drive. To do this, double-click on an icon for one of the hard disk's partitions. (If you haven't created icons for the partitions yet, do so. See below.) If the handler has worked properly, double-

clicking on a partition icon should bring up a window for the partition.

The manual method of hard disk booting isn't used very often, simply because it requires work on your part, and because it has to be done every time you turn on your computer.

However, I mention manual booting for two reasons. First, knowing that it's at least possible to manually boot your hard drive takes some of the mystery out of the booting procedure. Second, there may be times when you *would* like to use manual booting. We'll discuss just such a situation later on in this chapter.

2. Causing the handler program to be run automatically from a floppy. This is done by creating an AUTO folder on a floppy disk, copying your hard drive handler into the AUTO folder, then booting the computer with this disk in drive A. (Read your host adapter instructions and make sure to follow the steps in proper sequence. You'll probably have to turn on the hard drive first, wait for a few seconds, then turn on your ST.)

Unlike manual running of the handler, putting the program into an AUTO folder will cause the booting software to be run automatically every time you turn on your computer, provided the boot disk is in drive A when the computer is turned on.

3. Booting the hard disk from the hard disk itself. This method is usually called "autobooting." To autoboot from your hard disk, refer to the instructions for your hard drive software. (If you use ICD's HDUTIL.PRG, the booting program will be written to the root of C and renamed ICDBOOT.SYS. The boot sector of the disk will also be changed to reflect autobooting.)

Autobooting from the hard drive has the advantage of being quicker than booting from a floppy's AUTO folder.

DESKTOP ICONS

Even if your drive has been formatted, partitioned, and booted, you still won't be able to access the partitions from the Desktop with the mouse unless a disk icon has been "installed" for each partition. (The same thing holds true for a RAM disk or floppy

disk: if a drive doesn't have its own icon, there's nothing for you to double-click on.)

Installing a disk icon merely means creating an icon for a particular partition. Here's how to do it:

1. From the Desktop, click once on one of the existing drive icons. It doesn't matter which one you click on.

2. Choose INSTALL DISK DRIVE from the OPTIONS menu.

3. With the cursor on the DRIVE IDENTIFIER line, press the [Esc] key. This will erase the letter designation of the existing drive. Now type in the capital letter C. (Don't use the lowercase c as a drive identifier, since that letter is reserved for the cartridge port.)

   If your hard drive has only one partition, it should be designated C. If you have more than one partition, they should be assigned successive letters of the alphabet. For instance, if you have two partitions, they should be C and D; three partitions should be C, D, and E; etc.

4. Move the cursor to the ICON LABEL line, press [Esc], then type in a name for the drive. You can name partitions anything you want: "HARD DISK," "HARD DRIVE," "Programs," "Data," etc.

5. Mouse-click on the INSTALL box and a new disk icon will appear on the Desktop. If you don't see the new icon, it's probably hidden under a window or another disk icon. If so, you'll have to move the other icons and windows in order to uncover the new icon.

   You have now completed the installation for that partition's icon. However, if there are other partitions that do not yet have their own icon, repeat steps 1 through 5 for each remaining partition.

SAVING THE DESKTOP

In order to avoid having to go through the above icon-installation steps every time you turn on your computer, perform a SAVE DESKTOP operation. This will save your partition icons to the

DESKTOP.INF file, and will cause the icons to automatically appear every time you boot up. To save the Desktop, first make sure that all disk icons and windows are positioned on the Desktop the way you want them. Then choose SAVE DESKTOP from the OPTIONS menu.

TURNING THE COMPONENTS ON

If you've chosen to run your hard drive handler automatically (either from a floppy's AUTO folder or from the hard drive itself), the order in which you turn on the hard drive and computer is important. You must turn the hard drive on first, then wait until the drive gets "up to speed" before turning on your ST. Getting up to speed usually takes between 5 to 30 seconds, and after using your drive for awhile, you'll be able to tell by the sound of your unit when it's finished its powering up and unparking routines.

If, on the other hand, you want to manually run your booting software, it doesn't matter which component you turn on first. There's nothing to prevent you from turning on the computer, then turning on the hard drive, then running the handler.

When it comes time to shut down your system, it doesn't matter which component you turn off first: the hard drive or the computer. It's even all right to have them both plugged into a power strip and turn everything off all at once. Some people like to leave their hard drive on throughout the day, even when their computer is shut off. There's nothing wrong with this.

Finally, note that, depending on what type of host adapter you have, you may (or may not) be able to use just your ST without having your hard drive also turned on.

IMPORTANT WARNINGS

Here are some things that you should never do:

1. Don't treat your hard drive roughly, and don't try to move it when it's turned on. If it gets severely jarred, one of its heads may crash into a platter, resulting in permanent damage.
2. Although it's OK to look into the outer case that houses the

hard drive, controller, and power supply, don't ever open the hard drive mechanism itself. It's a specially sealed unit, and if any dust gets inside, reliability can be quickly destroyed.

3. At least one hard disk manufacturer has warned against trying to copy onto your hard drive any program that is copy-protected. Supposedly, some copy-protection techniques employ a "worm code" which destroys everything on the copied disk. If this happens to a floppy disk, all you've lost is the floppy. But if the worm goes to work on your hard drive, you could lose all the files on the drive.

4. Supra's manual for hard drives warns against performing a disk-to-disk copy of one partition to another. In other words, don't drag the disk icon for one partition onto another icon. Instead, to copy everything in a partition, first open a window for the source partition. Then mouse-select all the items in the partition and drag them to the target partition's icon.

5. Don't do anything that will restrict the air flow to your hard drive. Overheating is one of the worst enemies of electronic equipment, and if you back your drive into a corner, or pile papers on its ventilation holes, unpleasant things can start happening. If heat is allowed to build up around the controller and/or the hard drive mechanism, transient problems (which are the worst kind) can occur.

For instance, you can suspect that you've got a heat problem if you get messages from GEM saying that a drive is not responding, or that data may be damaged, even though you were able to access the drive a little while before, and access it afterwards when the system has cooled down.

6. Don't smoke in the same room where your hard drive is located. If even a small particle of smoke dust gets inside the drive mechanism, damage to the head and/or platter could result. And although hard drive mechanisms contain filtering devices, smoke in the air could conceivably clog the filters.

DESK ACCESSORIES

With a floppy-based system, any DAs (desk accessories) that reside on the root directory of your A drive boot disk are

automatically loaded into memory when you boot up.

But with a hard drive, you'll have to copy the accessories to the root directory of drive C. They'll load from the C drive regardless of whether your hard drive autoboots or boots from floppy. If there also are desk accessories on your floppy disk, they won't be loaded, provided there is at least one desk accessory on the C drive.

As discussed below, a corrupted desk accessory file can cause problems during the booting of your hard disk. For this reason, some users prefer to turn off their hard disk before trying out any new desk accessories. Once they've determined that the new accessory works properly on a floppy, they then feel safe copying it to the hard disk.

DESKTOP.INF FILE

When you turn on an ST that's connected to a hard drive, the computer configures the Desktop according to the information contained in the DESKTOP.INF file that's in the root directory of drive C.

However, if there isn't a DESKTOP. INF file in the root directory of drive C, the computer then looks to the root directory of drive A. If there's a DESKTOP. INF file there, then that file will be used to configure the Desktop.

If you've altered your DESKTOP. INF file from within a word processor or text editor, you might want to be on the safe side and try the file out first from a floppy. If you made a mistake in altering the file, trying it out on the hard drive could cause problems.

THE AUTO FOLDER

Using an AUTO folder to run programs automatically on bootup is just as easy with a hard drive-equippted ST as it is with a floppy-based system. However, with a hard drive, the question of *which* AUTO folder you use becomes important.

If you've chosen to autoboot your hard drive, only those programs residing in the AUTO folder on the root directory of drive C will be executed. If you also have programs inside the AUTO folder on drive A, they won't be executed.

Contrasted with this, if you boot your hard drive from drive A, all the programs in drive A's AUTO folder will be executed. If you also have programs inside the AUTO folder on drive C, they won't be executed.

THE 40 FOLDER LIMIT

There's a bug in the older versions of the ST's operating system that can create a problem for hard disk users. If you access too many folders, the operating system may get confused. If it does, the system can crash, or errors can occur during disk reads and writes.

Here's how the problem arises: when you double-click on a folder, not only does a directory window appear on your screen, but in order to save time, information about the folder is also stored in a buffer in the ST's memory. Storing this information saves time, since the next time the operating system needs information about that folder, it won't have to read the disk again.

If you double-click on a second folder, another memory buffer is filled with the second folder's information. As you access more folders, more buffers get filled up. The problem arises because, (a) there is a limit to the number of memory buffers that are available, and (b) backing out of hard drive folders (i.e., closing them) doesn't cause the space used by that folder to be returned.

With versions of TOS prior to version 1.4, the limit on the number of buffers is apparently somewhere around 40. Once all the buffers are filled, bad things can start happening. For instance, you may double-click on a folder and be told that it contains "0 BYTES IN 0 ITEMS," even though you know there are files in the folder. Other times, the system may merely crash.

But sometimes the symptoms aren't as innocuous. It's possible for the operating system to get so confused that it thinks it's writing to a particular file, whereas it's actually sending the data to a

different part of the disk. Obviously this could be disastrous, since any overwritten material would be destroyed. Data files, programs, or even whole directories could be lost.

IMPORTANT! If you get the "0 files in 0 items" message, re-boot immediately. If you don't re-boot, but instead write to the disk, the disk's directory structure will probably be damaged.

FORTY-FOLDER BUG SOLUTIONS

Luckily, there are a number of simple solutions to the 40-folder problem.

1. Occasionally turn the computer off, then on again (or press the Reset button). Turning the computer off clears out the buffers.

2. Use one of the programs which "fix" the 40 folder bug, such as FOLDRXXX.PRG. This program, written by Atari and freely distributed, raises the limit by whatever number you substitute for XXX in the program's name. For instance, if you rename the program FOLDR100.PRG, it will allow your system to access 140 folders without the limit being reached. (After you rename the program, copy it into the AUTO folder of whichever drive your system boots from, i.e., either the C or A drive.)

   Note, however, that FOLDRXXX.PRG is not a free lunch. The creation of more buffers results in slightly less available memory.

3. Acquire TOS version 1.4.

Here are some additional facts concerning the 40 folder bug.

1. The ICD and Supra booters automatically raise the 40 folder limit to around 100. Therefore, if your system uses a host adapter from one of these companies, you may not need to run a program such as FOLDRXXX.PRG. (In fact, the newest ICD boot driver is configurable: you can choose how many "folder slots" you'd like to have, up to 999.)

2. For all practical purposes, the 40 folder bug is confined to hard disk users. If your ST doesn't have a hard drive, it's

unlikely you'll be bothered by the problem. The reason is that, when you eject a floppy disk from its drive and then insert another floppy, the system senses that a new disk has been inserted. As a result, it clears out the old disk's buffers and puts in the new disk's directory information. Because of this buffer flushing, you can, for instance, feed as many floppy disks as you want into the A drive (and access all the folders on each floppy) and still not exceed the 40 folder limit, provided that no single floppy contains more than 39 folders.

3. When you're on the Desktop and you click once on a disk icon and then choose SHOW INFO from the menu bar, you will be accessing all the folders on that drive, even though you aren't specifically calling up windows for any of the folders.

For example, let's assume you have four partitions, each of which contains 30 folders. If you perform a SHOW INFO operation on all partitions, you will have accessed 124 folders. This is an example of how even those with Supra or ICD software can easily exceed their buffer limit.

4. Although the problem is usually referred to as the "40 *folder* bug," it would be more accurate to call it the "40 *directory* bug." Not only do you fill up buffers when you double-click on folders, but you also fill them up when you double-click on drive icons. What this means is that when you access your first folder, you've already filled up *two* buffers, since you needed to open a disk window in order to double-click on the folder.

Because of this, even if you have less than 40 folders on your drive, it's still possible to reach the limit. For instance, if your hard drive has two partitions (C and D), each of which has 19 folders, it's possible to reach the 40 folder limit (directories for C and D count for 2, plus 19 x 2 = 40). Of course, you'd have to access each of the 38 folders.

AVERAGE ACCESS TIME

One of the factors which determines how fast a hard disk will operate is how quickly, on the average, the disk can move one of its

heads from one track to another. This ability is measured in milliseconds, and is called "average access time."

You can easily check the average access time of a drive by running ICD's RATEHD.PRG. I ran this utility on my system and the program's printout of results is reproduced in Figure 6.2. (Note: RATEHD.PRG is copyrighted by ICD, so it's not in the public domain. However, ICD allows the program to be freely distributed. RATEHD.PRG will work in any ST system, regardless of whether you have an ICD host adapter or not.)

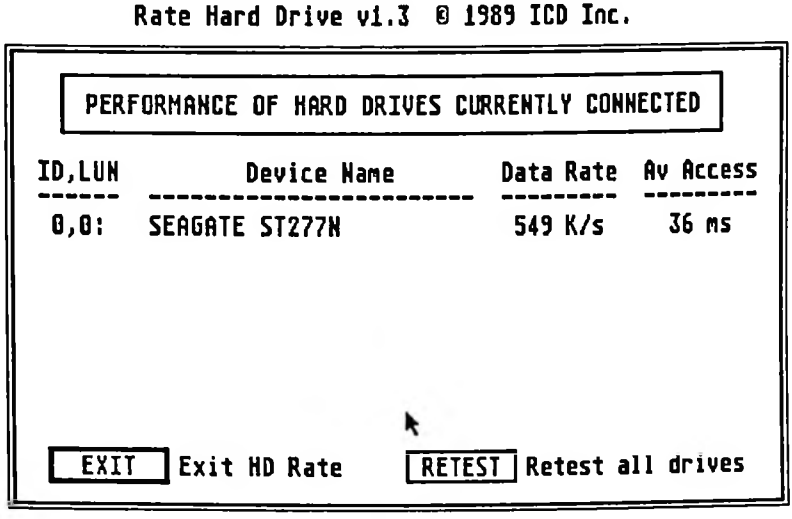


Figure 6.2

The lower the average access time, the better. Some drives are as low as 10 msec., which is very fast. At the other extreme, there are a few drives that are as slow as 105 msec.

INTERLEAVE

Another factor which affects disk speed is interleave. Interleave refers to how sectors are arranged on a disk (both hard and floppy

disks).

Interleave is expressed as a ratio, such as 1:1, 2:1, 5:1, etc. The interleave ratio represents the distance between two consecutively numbered sectors on a disk.

With an interleave of 1:1, consecutively numbered sectors are next to each other. For instance, sector number 2 is in between sectors 1 and 3. If the interleave ratio is 1:1, here's how the first 17 sectors on a track would be arranged:

1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17

If, on the other hand, a disk is formatted with an interleave of 2:1, then the distance between consecutively numbered sectors is two. In other words, consecutively numbered sectors are not right next to each other. With an interleave ratio of 2:1, here's how the first 17 sectors on a track would be arranged:

1-10-2-11-3-12-4-13-5-14-6-15-7-16-8-17-9

As you can see, sector 10 is in between sectors 1 and 2, sector 11 is in between sectors 2 and 3, and so on.

An interleave of 1:1 will result in the fastest data transfer your system is capable of. Changing to a 2:1 ratio would slow things down slightly, and changing to 3:1 would slow things down even more. Why?

With an interleave ratio of 1:1, the disk will have to rotate only once in order for one full track of sectors to be read (assuming the optimum interleave for your controller and host adapter is 1:1).

If, however, you change the interleave to 2:1, then your disk will have to rotate twice in order for all the sectors on a track to be read.

Although the lowest interleave ratio will, in general, result in the fastest performance, there's an exception. You don't want to choose an interleave ratio that's lower than the optimum rate of your controller/host adapter combination. For instance, if your system's optimum rate is 2:1, choosing a rate of 1:1 will cause the system to

be slower than choosing 2:1.

Would a person ever want to choose an interleave ratio higher than his system's optimum rate? Possibly, assuming he had a computer other than an ST. Whereas the ST's DMA (Direct Memory Access) chip, in conjunction with the host adapters and controllers used with an ST, allows a hard disk to be run at the very fast 1:1 interleave, not all computer systems are this efficient.

The original IBM XT, for instance, had to use a slower 6:1 interleave ratio when formatting. The computer's DMA channel and controller weren't capable of reading data quickly enough to keep up with the drive's transfer rate. By using an interleave of 6:1, the data transfer rate was slowed down to keep pace with the DMA channel and controller.

The interleave ratio is selected at the time of formatting. When you run your hard drive formatting software, an option will allow you to choose a specific interleave ratio.

How do you determine what interleave to choose? If you're using one of the more popular controllers, the interleave may already be chosen for you by your hard drive formatting software. If not, consult the manual for your particular controller (or drive, if the drive has an embedded controller). If the manual lists your controller's interleave capability, use that. If you don't have a manual for the controller, try calling the company that manufactured your host adapter.

The printout that accompanied my Seagate ST277N drive (with embedded controller) said, "interleave - 1." This meant that the controller was capable of a 1:1 interleave. (However, I've been told that, even if a drive's specifications indicate that it's capable of 1:1 interleave, this may not necessarily be the case. Supposedly, some companies fib.)

DISK OPTIMIZING

When you first use a hard drive, you'll be amazed at how quick it is. But as time passes, a strange thing will happen. Your drive will start to slow down. In fact, the more you use your drive, the slower it will get. The slowdown in speed will have been caused by a

number of things, one of which could be called "sluggish FAT searching." This phenomenon has to do with where the free space on your disk is located.

When you save a file to a disk, your computer sends the file to the first available cluster. The lower numbered clusters (i.e., those nearest the outer edge of the platter) get used first, then the clusters nearer the middle get used, and so on.

When your computer wants to save another file, it starts at the lowest numbered cluster and looks through the FAT (the file allocation table) to find clusters that are free. The more filled up a disk is, the longer it will take for a free cluster to be found.

Special programs called disk optimizers can correct this problem. Hard Drive Turbo Kit (by MichTron, 576 So. Telegraph, Pontiac, MI 48053. Ph. 313-334-5700) and Hard Drive Sentry (by Beckmeyer Development Tools, 592 Jean St., #304, Oakland, CA 94610. Ph. 415-658-5318) are commercial programs that, among other things, optimize files.

These programs can move all the free clusters to the beginning of a disk. As a result, the next time the computer wants to save a file, it very quickly locates free space in which to store the file.

The new ICDBOOT program allows configuring more memory for DOS FAT buffers, which reportedly eliminates the delay without messing around with the disk. TOS 1.4 also fixes the problem.

FRAGMENTATION

Fragmentation is another, although less serious, cause of speed loss. When you first use a newly formatted floppy or hard disk, there is no fragmentation. But as you begin to write, delete, and update files, individual files get split up into fragments. (Fragmentation was discussed in Chapter 5. See page 59.)

Here's how that happens. When a file is deleted, a gap is created on the disk where the deleted file used to be, and that gap now becomes available for use. If you then save a file that's larger than the old file, only part of the new file will fit into the gap left by the old file. As a result, the remaining part of the new file will have to be stored elsewhere on the disk. Part of the file will reside in one

sector, and another part will reside in another, possibly non-contiguous sector.

It takes longer for your computer to read a fragmented file than a non-fragmented file. The situation is analogous to the task of loading all your possessions into a moving van. If everything you own is located at one address, the job will go pretty quickly. But if your possessions are located at a number of different addresses, then the job of collecting them will take longer.

Fragmentation is a cumulative situation. As you save and delete more and more files, greater numbers of files become fragmented, and the degree of fragmentation of individual files becomes greater.

Although file fragmentation occurs on both floppy and hard disks, the amount of fragmentation that can occur on a floppy is limited by the relatively few number of files that can fit on the disk.

Program files may not be subject to the same degree of fragmentation that data files are. If you copy a program to a freshly formatted disk, that file will probably not be written to from then on. Data files, on the other hand (including data or system files that are associated with a program) are typically updated and written to disk frequently, and thus are more likely to become fragmented.

ELIMINATING FRAGMENTATION

There are a number of things that you can do to eliminate file fragmentation:

1. Use an un-fragmenting program. Such programs rearrange the files on selected partitions so that all the clusters of each file are contiguous. Hard Drive Turbo Kit (by MichTron) and Hard Drive Sentry (by Beckmeyer Development Tools), mentioned above, are commercial programs that un-fragment files. (Make sure and backup your hard drive before you try to un-fragment it.)

2. Backup your hard disk, reformat it, then copy the files back onto the disk. Although this takes a lot of time, the files become unfragmented in the process of copying them back

onto the disk.

3. Move all the files out of one partition and into another, then copy them back into the first partition. For instance, copy all the files in D partition into E partition, then erase all the files residing in D partition. Now copy the files from E back into D. As with method number 2, the copy process eliminates fragmentation.

CACHING

Another way to reduce the amount of time it takes your computer to load programs and access files is to use a disk caching program.

A disk cache is sort of a hybrid RAM disk. A caching program sets aside a portion of your ST's memory, then uses that area to store recently accessed portions of disk data.

To understand how this can speed things up, let's see what happens when you *don't* have a disk cache. When you load a program or access data from a hard or floppy disk, here's what normally happens:

First, the operating system looks at the disk directory in order to find out where the beginning of the file is located. Once it finds that out, it goes to that spot on the disk, then waits for the first sector of the file to come by. Then it reads that first part of the file into memory.

Often, however, the file may be fragmented, which means that the rest of the file is scattered throughout the disk in many fragments. It will take a certain amount of time for the operating system to move around and pick up the pieces of the file.

If you have a disk cache, however, the situation is changed. The first time the OS (operating system) reads data from a disk, it does two things. First, it stores the data in the cache, and second, it sends the data to whoever requested the data (a program, GemDOS, etc).

The next time the OS receives a request to access data from the disk, it first looks into the cache's memory buffers to see if the

requested data is there. If that data has been requested before, it may be in the cache. And if it is, the OS will be able to quickly copy it from the cache, and it won't have to go to the disk to read the data.

The reason that a disk cache speeds things up is that it takes far less time to read data from part of the computer's memory than it does to read it from a disk.

In addition to speeding up the loading of data from disk, a disk cache may also speed up the writing of data to disk. Speedier disk writes occur because the directory for a disk is also stored in the cache. It's much quicker for the operating system to search a directory that's located in memory than one that's on a disk.

The newest ICDBOOT program contains a configurable cache. L-CACHE.PRG is a shareware disk caching program created by Keith Ledbetter. MichTron's Hard Drive Turbo Kit also contains a caching program.

Although a disk cache can be very useful, its benefit will be appreciated primarily in programs that are very disk intensive. For example, data base managers, compilers, and accounting programs, all of which frequently access data from disk, are likely to benefit most from a disk cache. Contrasted with this, if you use a program that loads itself completely into memory and doesn't spend much time reading or writing data to disk, you may not find caching particularly helpful.

PARTITIONING FOR SPEED

Here are two ways of using partitions to improve performance of your hard drive:

1. Partition your drive so that there is at least one relatively large partition, as well as one quite small one. Keep your programs and infrequently used data files in the large partition and reserve the small partition for your "active" data files (i.e., those that you use most often).

   Since each partition has its own FAT table, the FAT table for the small partition will be quite short. As a result, when the operating system looks for one of your frequently used files, the

search will be performed very quickly.

2. The more you put on a disk, the slower it gets. In fact (under the older versions of TOS) a full disk may take nearly ten times as long to duplicate a file as an almost empty disk. (TOS 1.4 speeds things up a lot.)

One way to deal with this problem is to create partitions. Partitioning results in a number of smaller "drives," and since each resultant drive contains less data, the speed of operations increases.

In choosing partition sizes, you'll have to take into consideration two opposing factors. On the one hand, smaller partitions result in greater speed. On the other hand, smaller partitions may not be able to hold large collections of related files, and the maze of icons on the Desktop could be confusing. Partitions that are about 5 megabytes in size seem to be common.

LOCKOUT

If you've ever locked your car with the keys inside, you know how frustrating it can be. A similar situation can happen with your hard drive.

It's possible for something to go wrong during the booting process that results in you being "locked out" of your hard drive. In other words, you can't access the drive. Depending on what's causing the problem, the symptoms can vary. If the drive can't boot, it may get stuck in a cycle of resetting and trying to boot, over and over again. In some situations, the whole system may just lock up during the booting process.

Listed below are some of the situations that can cause hard drive lockout.

1. Having a corrupted desk accessory on the root directory of your hard drive.

2. If you auto-boot your hard drive, having a corrupted or bug-ridden program inside the AUTO folder on your hard drive's C partition.

3. If you boot your hard drive from a floppy, having a corrupted or bug-ridden program inside the AUTO folder on your A drive.

3. Having a corrupted DESKTOP.INF file in the root directory of your hard drive's C partition, or in the root directory of your A drive boot disk.

4. Having corrupted hard drive booting software on drive C, or in the AUTO folder of your A drive.

WHAT TO DO

If you can't get your hard drive to boot properly, try one or more of the following techniques.

1. If your system was set up for autobooting, read the instructions for your hard drive booting software to see if it's possible to circumvent autobooting. (The software directions may instruct you to make sure your booting program is in the AUTO folder on drive A. Then turn the power to your ST on while holding down the [Shift], [Alternate], and [Control] keys. However, with TOS 1.2 and 1.4, you must wait for the floppy's light to come on before pressing the [Shift]+[Alt]+[Control] keys.)

2. Try manual booting. Regardless of whether your system was set up for autobooting or for booting from floppy, here are the steps:

a. Leave the hard drive connected to your ST.

b. Turn the power to your hard drive off.

c. Make sure that whatever disk is in drive A does not contain an AUTO folder or any .ACC or .INF files.

d. Turn your ST on.

e. Turn on the power to your hard drive; then wait for it to get up to speed.

f. Insert into the A drive a disk containing your hard drive booting program (ICDBOOT.PRG, SUPBOOT.PRG, AHDI.PRG, etc.), then run the program by double-

clicking on it.

g. Install a drive icon for C.

h. Double-click on the C icon. If you're lucky, a window for the C partition will open. If so, you've successfully accessed your hard drive.

3. If neither manual booting nor the "Shift/Alternate/Control" key-combination is successful, contact the manufacturer of your host adapter (Supra, ICD, Berkeley MicroSystems, etc.). They may be able to suggest an additional procedure to try.

4. Try HD–WEDGE, a public domain program which reportedly allows you to wedge yourself into a hard drive from which you've been locked out.

You could also try REVIVE.PRG, which is similar to HD–WEDGE.ARC. REVIVE.PRG was on the disk edition of the spring 1987 issue of STart Magazine.

CORRECTING THE PROBLEM

Once you've accessed a locked-up hard drive by using one of the above-mentioned methods, you've still got to correct whatever caused the problem. Here's how to use the process of elimination to track down the offending .ACC, .INF, or .PRG file that created the problem in the first place.

1. Rename all the .PRG files that reside in the AUTO folders on drive C and A. If you change these files' extenders to something like .PR, the programs won't be run the next time you boot up.

2. Rename all the desk accessories on the C drive's root directory. For instance, you can change their extenders from .ACC to .AC. This will prevent them from being loaded into memory the next time you boot up.

3. Rename any DESKTOP.INF file that's on your C drive's root directory, or on your boot disk.

4. When there are no "active" .INF or .ACC files and no "active" .PRG files inside an AUTO folder, boot your hard drive in whatever way you normally do: either by autobooting or

booting from floppy. If the drive boots properly, proceed to step 5.

If the drive doesn't boot properly, your hard disk booting software may be corrupted. Replace the copy of the program with a fresh version from the master disk you received when you bought your drive.

5. Change the name of one of your .ACC, .INF, or .PRG files back to what it used to be. For example, pick one of the desk accessories and change its extender back to .ACC. Now reboot. If the booting fails, then this file was probably the cause of the problem. Trash the file, as well as any other corrupted copies of it that exist.

If, on the other hand, the hard drive booted properly with this file, then this file wasn't the cause of the problem. In this case, proceed to test each remaining .ACC, .INF, and .PRG file in the same way, one file at a time (re-booting in between each try). You should eventually find the file that's been causing the problem.

In using the process of elimination to track down a corrupted file, keep in mind that the culprit is more likely to be a new desk accessory or autorun program, rather than files which have worked without problem in the past, such as CONTROL.ACC. However, it's also possible for two otherwise well-behaved programs to interfere with each other. You may have to do a lot of experimenting to track down this sort of problem.

HEAD PARKING

Among the programs that came with your hard drive was probably a head parking program. Such a program moves a drive's heads from the data area of a platter to an unused part of the disk. It's advisable to run one of these programs before you transport your drive, since rough handling of the drive can cause an unparked head to crash into the platter, resulting in permanent damage to the platter's surface.

Some users even believe that merely shutting off the hard drive

can be dangerous, since the heads will come to rest on the data area of the platters. To be safe, these users always run their park program prior to shutting down their system.

Nowadays, many drives are "self-parking," which means they automatically move their heads to a safe area whenever they're powered down.

BACKING UP

With a hard drive, backing up becomes much more important than with a floppy-based system. This isn't because hard drives are more unreliable -- they aren't -- it's just that more of your eggs (possibly all of them) are now in one basket. If your drive suffers a major crash, you could lose all your files.

There are a number of ways to backup the files on your hard disk.

1. Using the regular Desktop copy method, you can copy files, one file at a time, to a floppy. This works nicely when (a) each individual file is not too big to fit on a floppy, and (b) you have a small number of files to backup.

   I use this method often. For instance, after I've finished a day's work of writing, I close my word processor, then drag a copy of the data file from the hard disk to a floppy.

2. If you have so many files to backup that the "one file at a time" method isn't practical, you can use one of a number of programs designed specifically for backing up hard drives.

   TURTLE, a public domain program by George Woodside, is one such utility. It can be used to backup a whole disk, a selected folder or folders only, or just selected files. The two major limitations of TURTLE are that (a) it requires a minimum of 1 megabyte of memory, and (b) it's not capable of backing up a file that's too big to fit on one floppy disk.

3. Neither of these limitations is a problem for MEG-A-MINUTE ELITE, by Dave Small. This program appeared on the disk version of the November 1988 issue of STart magazine, and is available from Antic publishing.

4. For backing up files that are too large to fit completely on one floppy, George Woodside has created TERRAPIN, which is also in the public domain. TERRAPIN splits up a large file into segments, then copies the segments to multiple floppy disks.

5. If you have the money, purchasing a second hard drive can make backing up a snap. Just copy the contents of one drive to the other. (But remember, don't copy disks using the icon-to-icon method.) The nice thing about using a second hard drive for backup is that you don't have to keep feeding floppy disks into your drives.

6. There also are a number of units made specifically for backing up hard drives. One such system, ICD's tape backup unit, reportedly copies a 15 megabyte partition in a couple of minutes.

USEFUL PROGRAMS

Here are just a few of the many programs that can make using a hard disk easier.

1. A replacement for the Item Selector. With a hard drive, you'll have more icons on the Desktop. If you're using the old GEM Item Selector, accessing files in different drives can be annoying. The following two programs both provide an "immediate-choice-of-drive" capability. To change from a display of the contents of one drive to that of another, all you do is mouse-click on the drive letter you want.

The Universal Item Selector, a commercial product, is available from Application & Design Software, 226 NW "F" Street, Grants Pass, OR 97526. PH: 503-476-0071. Among other features, the program allows you to: set up item selector defaults; shift-click or rubber band groups of files; format drives; create folders; find files on any drives; move files or folders; copy files; rename files or folders; set file attributes; print files; and show files to the screen. (See Figure 6.3)

The Little Green Selector, a sharware program by Charles F Johnson, is also a fine replacement for the old Item Selector. (See Figure 6.4)

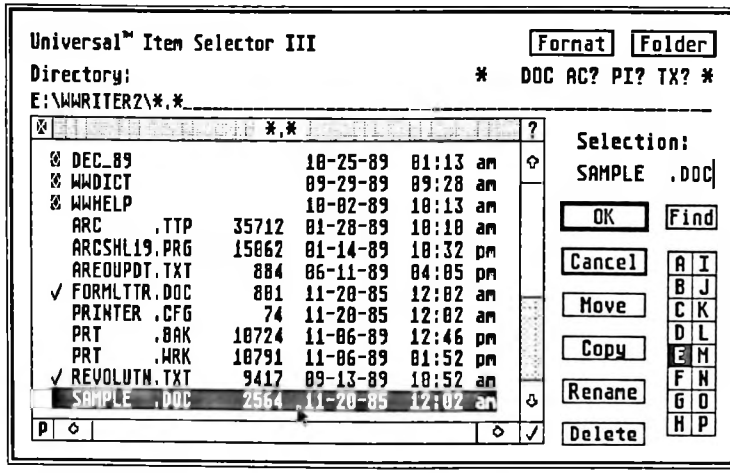


Figure 6.3

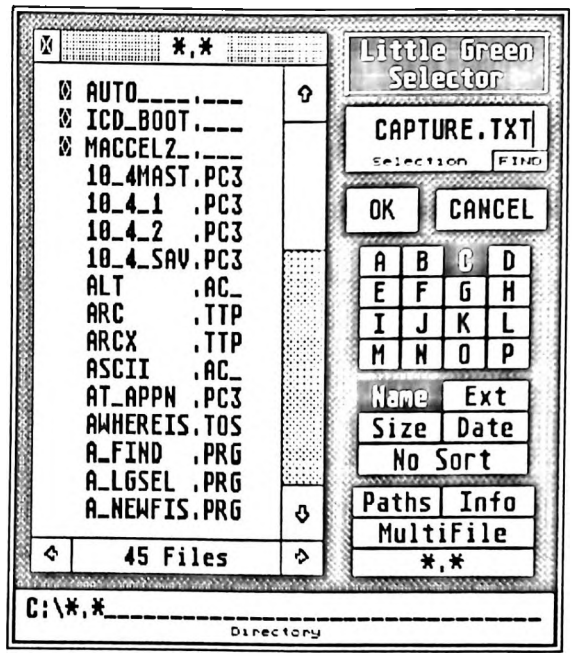


Figure 6.4

2. Disk structure testers. With time, the data on disks can become corrupted. MichTron's Hard Drive Turbo Kit contains a program that checks to see if the disk directory correctly represents what's actually on the disk. If it doesn't, the program lets you correct the problem.

ICD's CLEANUP program does a particularly thorough job of correcting disk structure corruptions. It is claimed that CLEANUP repairs errors that have been caused by:

- poor software
- power surges and power failures
- deterioration of the magnetic coating on the disk platter
- a faulty disk controller or DMA chip
- failing to re-boot after a program bombs
- pressing [Reset] while a program is writing to the disk
- virus programs

NOTE: CLEANUP will only work if you have an ICD host adapter in your system.

3. File finders. The more files you store on a disk, and the more folders you create, the harder it becomes to find a particular file. Among the many good programs to help locate files are Supra's SUPFIND.PRG and ICD's WHEREIS.TOS. (See page 148 for a description of the shareware program QUICKFIND, a superb file finder.)

Chapter Seven

USING MULTIPLE HARD DRIVES and ASSEMBLING A HARD DRIVE SYSTEM

QUESTION: A friend told me that you can buy a hard drive for $75 and hook it up to the ST. Is this true?

ANSWER: Yes. But your question reminds me of a conversation I had with my father a few years ago. "Hey, Junior," he said. "Did you read that they're selling a $1,000 electric car that costs only two cents per mile to operate?" "Really, Dad?" I said. "Yup," he continued. "The only problem is, the extension cord costs $20,000."

Connecting a $75 hard drive to your ST is a little like that, since a hard drive mechanism, by itself, can't be directly hooked up to an ST. A number of other components are needed in addition to the actual hard disk mechanism. Also, although it's possible to purchase a drive mechanism for $75, you might be better off buying a bigger and newer drive than those that can be purchased for $75.

However, if you think you've got the skill to tackle the job, you may be able to save some money by assembling your own system.

Just how much you'll save will depend, in part, on how large a drive you want. If all you need is a single 20 megabyte unit, then you probably won't save much money. But if you need really large storage capacity, or you want multiple hard drives, constructing your own system can produce significant savings.

But before I tell you about the specific steps I took in order to assemble my own system, there are a few general principles that we should cover.

SCSI AND LUN NUMBERS

As we've seen, a hard drive system for the ST usually consists of three components: host adapter, controller, and hard drive mechanism. (See Figure 6.1 on page 75)

However, it's also possible to create a system that contains more than just these three basic components. For instance, a system could consist of a host adapter, plus a controller that's connected to two (or more) hard disk mechanisms. Alternately, you could have a host adapter that's connected to two (or more) controllers, each of which is connected to its own drive(s). And finally, it may be possible to have more than one host adapter, each of which services its own controllers and drives. (See Figure 7.1)

Let's consider each component individually, starting with the host adapter. A single cable, using 16 signal wires and fitted with DB-19 (19-pin) connectors on both ends connects the host adapter to the DMA port on the back of the ST. The signals travel from the ST to the host adapter, where they are converted into signals that conform to the SCSI standard. (If your host adapter has a second DMA port, you may be able to daisy chain additional DMA devices, such as other host adapters or "stand alone" hard drive units, together.)

Once the signals have been translated into the SCSI standard, they leave the host adapter through a "SCSI bus" (i.e., a cable). The SCSI bus usually consists of a 50-pin ribbon cable, from which the SCSI devices tap off.

The most common type of SCSI device is a hard disk controller, but other devices, such as tape backup units, CD-ROM drives, etc., can also be plugged into the SCSI cable. Although the ST is capable

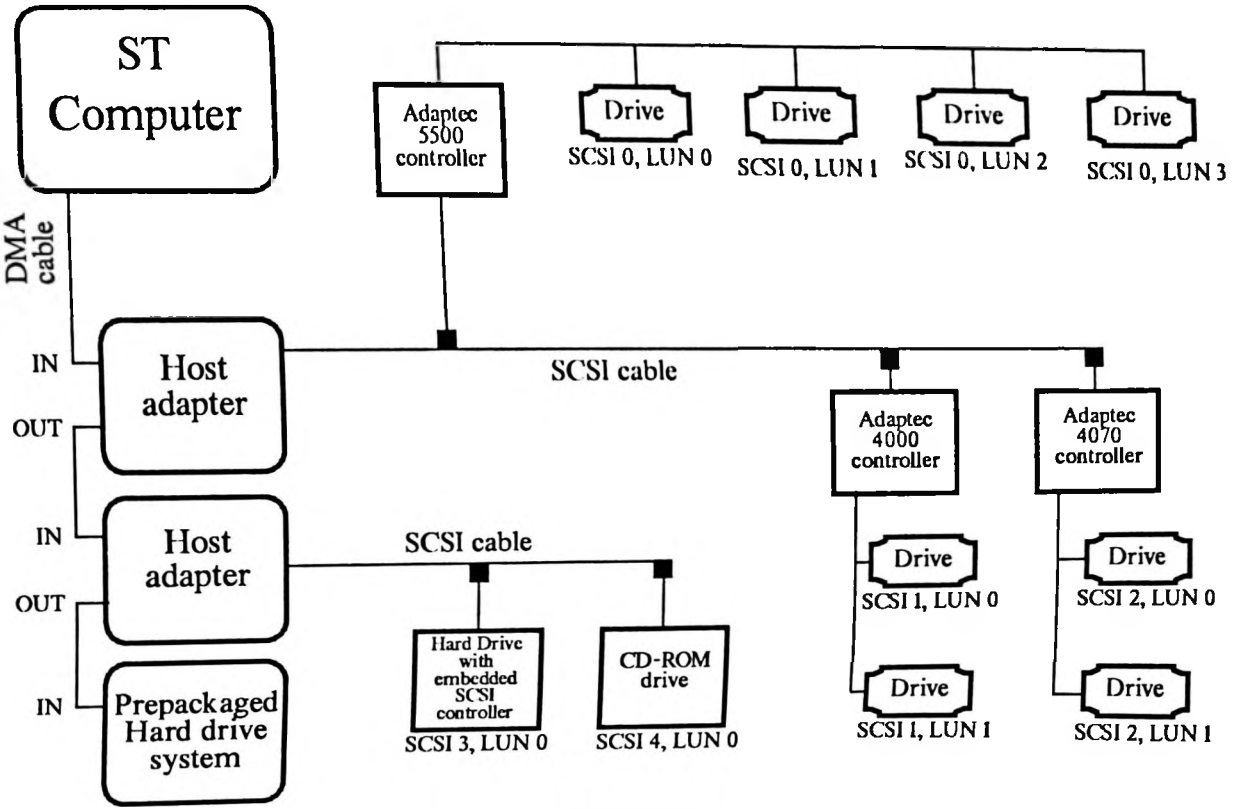

ST Computer
DMA cable
IN
OUT
IN
OUT
IN
Host adapter
Host adapter
Prepackaged Hard drive system
Adaptec 5500 controller
Drive
SCSI 0, LUN 0
Drive
SCSI 0, LUN 1
Drive
SCSI 0, LUN 2
Drive
SCSI 0, LUN 3
SCSI cable
SCSI cable
Hard Drive with embedded SCSI controller
SCSI 3, LUN 0
CD-ROM drive
SCSI 4, LUN 0
Adaptec 4000 controller
Drive
SCSI 1, LUN 0
Drive
SCSI 1, LUN 1
Adaptec 4070 controller
Drive
SCSI 2, LUN 0
Drive
SCSI 2, LUN 1

Figure 7.1

of handling up to eight different SCSI devices, this number may be limited by the type of host adapter you have, and by whether or not any of the host adapter's SCSI ID addresses are already being used. (For example, the ICD host adapter uses SCSI ID number 6 to run its real-time clock.)

Typically, 50-pin female connectors are crimped onto the SCSI cable. One of these connectors gets pushed onto the 50-pin male header plug that's located on the host adapter, and the remaining connector(s) on the cable are pushed onto 50-pin male header plug(s) that are located on the SCSI device(s).

Since the SCSI bus permits more than one SCSI device to be tied into it, there needs to be a way for the ST to keep track of which SCSI device it's communicating with at any given time. This is done by assigning SCSI ID (identification) numbers to the different devices connected to the SCSI cable.

SCSI ID number "0" is assigned to the first SCSI device on the cable. If you then want to connect another SCSI device, that second device will be designated SCSI ID number "1". Additional SCSI devices will be given ID numbers "2", "3", and so on.

Once you've determined what SCSI ID number will be assigned to a particular controller, you'll have to make some sort of physical adjustment on the controller board so that the appropriate ID number is internally selected in the controller. *What* it is that you'll have to do to make the setting may vary, depending on the design of the particular SCSI device.

For example, on Adaptec controllers, the SCSI ID selection is made by inserting one or more jumper plugs (which are shorting blocks) over selected pairs of metal jumpers. Other controllers, however, may use a different method of making the selection. To determine how to make the SCSI ID selection for a given controller, consult the manual for the controller, or talk to the technical support staff at the company that made the controller, or at the company that manufactured your host adapter.

CONNECTING MULTIPLE DRIVE UNITS

So much for controllers. But how do you connect multiple drive mechanisms? In most ST hard drive systems, controllers and drive

mechanisms communicate with each other using the "ST506/ST412 interface." Unlike the single SCSI cable between host adapter and controller, the ST506/ST412 interface employs *two* cables between controller and drive. (If your drive has an embedded SCSI controller, there won't be any cables between the controller and drive, since these two components form one integral unit. The controller/drive unit will connect directly to the 50-pin SCSI bus ribbon cable that comes from the host adapter).

To hook up the cables between a controller and its drive(s), refer to your controller's manual. With the popular Adaptec 4000 controller (which uses the ST506/ST412 standard) a 20-pin flat ribbon "data" cable runs from the "J0" connector on the controller to the first drive mechanism. If you have a second drive, another 20-pin data cable will run from the "J1" connector on the controller to the second drive. Each 20-pin cable uses an edge connector at the drive end and a header socket at the controller end.

Additionally, a 34-pin flat ribbon "control" cable runs from the "J2" connector on the controller to both drives. If you have only one drive connected to this controller, the 34-pin cable will have two edge connectors: one for the controller, the other for the drive. If you have two drives connected to this controller, the 34-pin cable will have three edge connectors: one for the controller and one for each drive.

LUN IDs

When more than one hard drive mechanism is connected to a single controller, a method is needed that will allow the ST to keep track of which drive it's communicating with at any given time. This is done by assigning a unique LUN ID (logical unit number identification) to each drive mechanism.

LUN ID "0" is assigned to the first drive mechanism; LUN ID "1" is assigned to the second drive mechanism, and so on. (The maximum number of drive mechanisms that can be connected to one controller will be determined by the type of controller you use. Consult the controller's manual for its specifications.)

As was the case with SCSI ID numbers, once you've determined

what LUN ID will be assigned to a particular drive unit, you'll have to make a physical adjustment on the drive so that the appropriate LUN ID is internally selected in the drive's circuitry. Typically, jumper plugs will have to be inserted (or removed) from selected pairs of metal jumpers. You'll have to refer to the manual for a particular drive in order to determine how to make the selection.

Finally, if you've installed more than one controller and/or more than one drive unit, you may have to remove "resistor termination packs" from (a) all controllers except the controller that's last, physically, on the SCSI bus, and (b) all drive units except the drive unit that's last, physically, on its controller's 34-pin control cable. (If you've installed only one controller and one drive unit, the resistor termination packs should probably *remain* installed on both controller and drive.) Refer to your controller and drive manuals for specific resistor termination pack instructions.

MFM AND RLL

MFM (Modified Frequency Modulation) and RLL (Run Length Limited) are terms that refer to the density scheme employed by a drive and its controller. Since a controller and its drives must use the same density scheme (either MFM or RLL), you'll want to determine whether a potential component is MFM, or RLL, before you team it up with its counterpart (either controller or drive). Coupling an MFM drive with an RLL controller (or vice versa) will jeopardize data integrity, as well as void both units' warranties.

RLL is the newer technology, and packs one and a half times as much data onto the same amount of space as MFM (i.e., more data per track). As a result, an RLL drive/controller combination can transfer more data before it has to move the drive's heads, and this results in quicker access time. (The data transfer rate for RLL is 7.5 megabits per second; for MFM, it's 5.0 megabits per second.) The megabyte cost per dollar is usually less with RLL than with MFM.

On the other hand, MFM controllers often cost slightly less than RLL controllers. Additionally, if you're interested in looking for used drives, many of the older MFM drives can be bought very cheaply. (I recently purchased a perfectly good, 40-ms average

access time 10 megabyte drive for $37.50.)

Now for a few cautionary words.

This chapter does *not* contain comprehensive instructions for hooking up hard drives. The specifics of how to assemble particular components will vary, depending on many factors: type of drive used, type of controller, size of case, etc. The only thing I do in the following pages is tell you how I put together *my* particular system.

Before you consider attempting a project of this nature, do the following:

1. Be aware that modifying any of your existing hardware will void its warranty. Even opening the case to look at an existing drive may void the warranty.

2. Contact the manufacturer of the host adapter you plan to use (Supra, ICD, Berkeley MicroSystems, etc.) and ask if the controller, hard drive, host adapter, and other components you want to buy will work properly together.

3. Make sure you know how to safely handle exposed electronic parts. Static electricity from your hand, for instance, can ruin a $300 component in a fraction of a second.

4. Understand that assembling a hard drive system is not a project for the novice. Although it may not involve any soldering, and although no complicated or sophisticated tools are required, it *is* a risky undertaking. If, when you get your system assembled, it doesn't work, it can be very frustrating.

If you've purchased your components from different suppliers, as I did, it'll be easy for each supplier to say, "The problem is with the other supplier's unit." Because of this danger, it might be best for many of you to play it safe and purchase an already-assembled system.

Note that, even for those of you who do decide to purchase an already-assembled unit, this chapter will still prove useful by increasing your understanding of hard drives.

MY FIRST SYSTEM

Here are the parts I used in constructing my first ST hard drive system, along with their cost:

- Seagate ST277N 60 meg hard drive (with embedded controller)	$ 391
- ICD host adapter	$ 95
- ICD mounting bracket	$ 9
- 150-watt power supply	$ 36
- IBM-style XT slide type case	$ 26
Total	$ 533

I bought the ICD host adapter and bracket from an Atari mail order dealer after leafing through a number of Atari magazines. Then, after looking at the ads in *Computer Shopper Magazine*, I bought the remaining components from three different dealers.

PUTTING THE PARTS TOGETHER

ICD host adapters come with instructions that cover their installation, formatting, and trouble-shooting. In the material that follows, however, I'll be describing in detail how I installed the

system's components into the case. And although I chose to use an ICD host adapter, the following account should generally be applicable no matter what brand of host adapter you buy. However, before you start assembling your own system, make sure to read thoroughly the instructions that came with your host adapter.

My first step was to slide off the cover from the IBM case. To do this, five screws had to be removed from the back of the case. The power supply, which fitted into the back right-hand corner of the case, then had to be secured to the case with screws. (Months later, after I had acquired more hard drive mechanisms than would fit into the single case, I bought a second IBM case. This time I bought a "flip-top" case, which is easier to open.)

I decided to attach ICD's mounting bracket to the back of the IBM case. This was the hardest part of the whole project, and the only part that required any drilling or decision making. (I would have avoided the drilling if I had bought an ICD case, instead of the IBM-clone case. ICD cases are designed to accept the ICD mounting bracket.)

The ICD mounting bracket consists of an "L" shaped piece of metal. One leg is 5 inches long, the other is $^1/_2$ inch wide. I planned to bolt the narrower leg to the back of the IBM case, and the host adapter to the 5-inch side of the bracket.

In the back left-hand corner of an IBM case you'll find a number of expansion slots. My case had five of these, each one of which was covered with its own slot cover. Since my Seagate 277N drive mechanism had an embedded SCSI controller, it didn't matter which slot I installed the host adapter in. All the cables were long enough to allow the host adapter to reside in the port on the far left-hand side.

I removed one of the port covers by unscrewing its securing bolt, then held the host adapter's mounting bracket (without the host adapter attached to it) up to the open slot. With a pencil, I marked where the bracket's two mounting holes met the back of the case, then made drill marks. The holes were actually drilled in the adjacent slot cover, not in the IBM case. This made the job easier, since I could remove the slot cover and clamp it (along with a piece of wood to prevent it from bending) in a vice for drilling. Once the holes were drilled, I used a file to remove a few metal burs. Then I

attached the slot cover to the mounting bracket, using a couple of rubber grommets as spacers, and two bolts that came with the bracket. (See Figure 7.2)

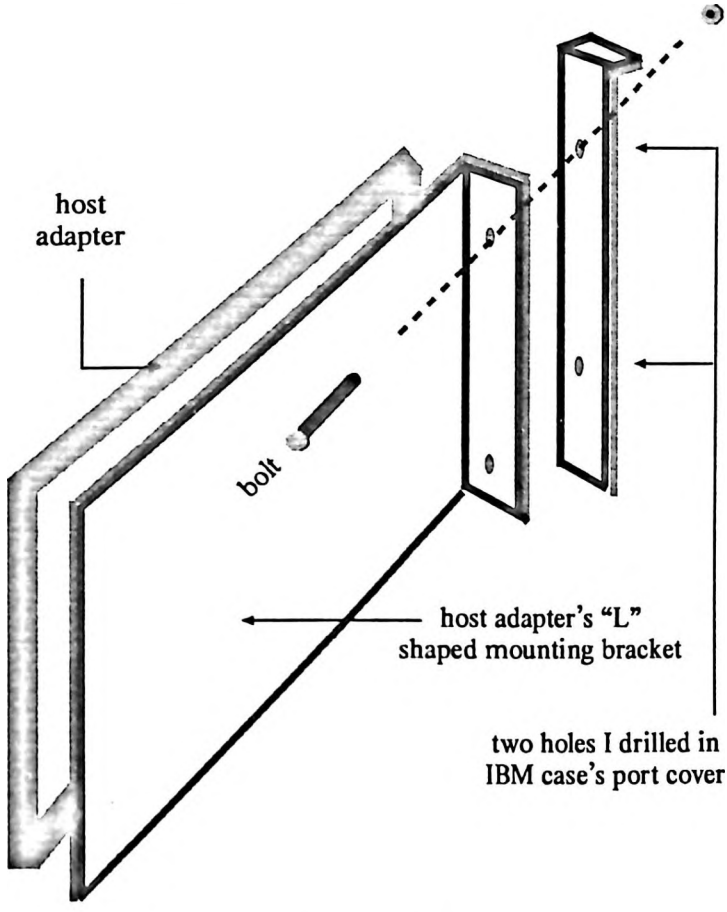


Figure 7.2

There were two advantages of this installation design. First, since the host adapter's ports line up with the XT case's open port, I can remove and attach the cable from the host adapter without having to open the XT case. Second, if I want to remove the host adapter, all I do is unscrew the single bolt that secures the port

cover that the host adapter's mounting bracket is bolted to.

Now it was time to attach the host adapter to the mounting bracket. Doing this was pretty straight-forward, except that I had to make sure to separate the adapter and its mounting bracket so they didn't touch each other. If they touched, they could short out.

To isolate them, I placed fiber washers on each of the four bolts that secure the adapter and the bracket. Actually, it's not enough to use washers *between* the two components. I also had to use washers between the heads of the bolts and the surface of the host adapter that faces the bolt heads. In other words, both sides of the host adapter should be protected with washers. Since there are four mounting bolts, I needed eight washers.

ICD's host adapter comes with a power lead, one end of which plugs into the adapter. The other end plugs into the power supply. But since it looked as if it might be difficult to attach the lead to the host adapter once the adapter was mounted to the case, I connected the power lead to the host adapter before mounting the adapter to the case.

The power lead is necessary because the ICD host adapter doesn't accept a standard IBM power plug. One end of the ICD power lead has a big plug, the other end has a small one. I pushed the end with the big plug into one of the leads from the IBM power supply. The end with the small plug got connected to a receptacle that's located in the middle of the bottom edge of the host adapter.

Next, I bolted the mounting bracket to the port cover, then screwed the port cover to the IBM case. I was now ready for the next step: installing the Seagate hard drive.

SETTING THE DRIVE'S JUMPERS

Before I could install the drive, however, I needed to make sure that the proper SCSI ID address was selected. Since the Seagate 277N drive/controller was the only SCSI device that would be connected to the host adapter, I needed to set the drive's jumper so that the drive would be recognized as SCSI device number "0".

On the Seagate 277N, if none of the jumper pins are shorted out, the drive will be recognized as SCSI device number "0". (If, on the other hand, two particular pins are shorted out with a jumper, then the drive will be recognized as SCSI device number "1". To determine which pins to short out, refer to the manual that came with the 277N drive.)

The 277N manual also instructed that, if the drive was going to be the only SCSI device in the system, I should leave the resistor termination pack in place. As before, I had to refer to the manual's diagrams and instructions. Since the 277N has an embedded controller, there was no LUN identification selection to be made.

The bottom of the Seagate ST277N drive consists of a printed circuit board which I didn't want to touch with my bare hands. To avoid this, I kept a piece of cardboard between my hands and the drive when holding the drive and bolting it into the IBM case. I had chosen to place the drive in the upper right bay of the case, and it was an easy matter to screw in the four bolts that held the drive in place. (The bolts came with the case.)

Connecting the power to the drive was easy, too, since the 277N, like most drives, has a power receptacle that matches the power leads from standard IBM power supplies.

THE SCSI CABLE

The dealer that sold me the Seagate ST277N had included a 50-wire ribbon cable in the sale, but I could have bought one from ICD, SUPRA, Berkeley MicroSystems, or from most large computer suppliers. I also could have made one myself. Attaching the ribbon cable was easy. There is only one 50-pin connector on the ICD host adapter, and only one 50-pin connector on the Seagate ST277N. However, I had to make sure that whenever I pushed a female connector onto a male header plug, I was connecting pin "1" on the connector to pin "1" on the header plug.

The connectors that you crimp onto ribbon cable usually have an arrow molded onto their side. This arrow indicates which end of the connector contains pin "1." On ribbon cable, one of the wires is usually identified by a different color, often red. (In other words,

there is a stripe on one side of the cable.) Position the "stripe side" of the cable so that it lines up with the "arrow side" of the female connector. Before pushing the connector onto a controller board (or onto a drive) determine which end of the board's header plug contains pin 1. The board will usually have a "1" (or a "2") printed near one end of the header plug. This is the end that should receive the pin 1 side of the female connector.

Next I had to connect the host adapter to my ST. There are two 19-pin jacks on the ICD host adapter board. If you look at the board closely, you'll see the words "IN" and "OUT" printed near the jacks. The ICD host adapter comes with a 3-foot long cable. One end plugs into the DMA port on the back of the ST, the other end to the "IN" jack on the ICD host adapter.

At this point, all that remained to be done was to recheck my work to make sure I hadn't made any mistakes. I then turned on the switch for the IBM's power supply, and was greeted with the pleasantly weird powering up sound of my new drive.

FORMATTING THE HARD DISK

In the same way that, before you can use a brand new 3 ½-inch floppy disk you have to format the floppy, a hard disk has to be formatted before it can be used. The steps involved in formatting will vary, depending on what company's formatting software you're using. Therefore, you'll have to follow the instructions that came with your hard drive or host adapter. However, here are some general suggestions.

1. If your formatting software will allow you to print out a list of bad sectors discovered during formatting, make sure that before you start the formatting process your printer is connected to your ST and is ready to print. If the printer isn't already connected, be sure to turn it, and your ST, *off* before connecting the cables. If the power is on when you connect the printer to the ST, you could ruin your ST.

2. If you have more than one hard drive connected to your ST, you'll be on the safe side if you disconnect all but the drive that you want to format. If this isn't done and you make a mistake

when setting up the formatting, you could end up formatting the wrong drive, which would erase all its data.

If you decide *not* to disconnect additional drives, scrutinize very carefully the information from your formatting software that appears on the screen. Make sure that the drive listed is the one that you want to format. (Figure 7.3 shows a screen from the ICD formatter.)

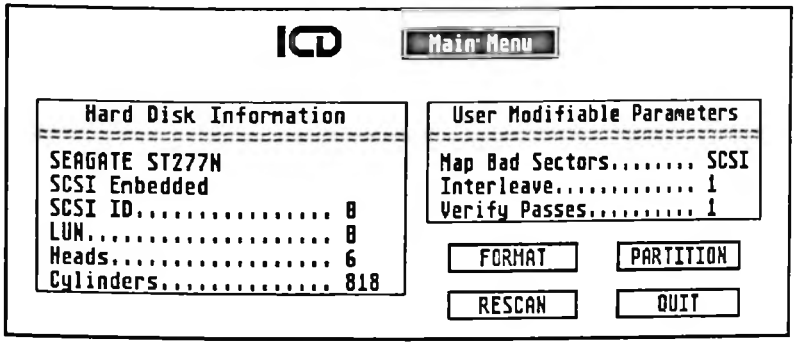


Figure 7.3

3. After following your formatter's instructions for selecting the appropriate controller and drive information, you may have to choose the appropriate SCSI ID and LUN numbers for the drive. (Figure 7.4 shows a screen from the Supra formatter.)

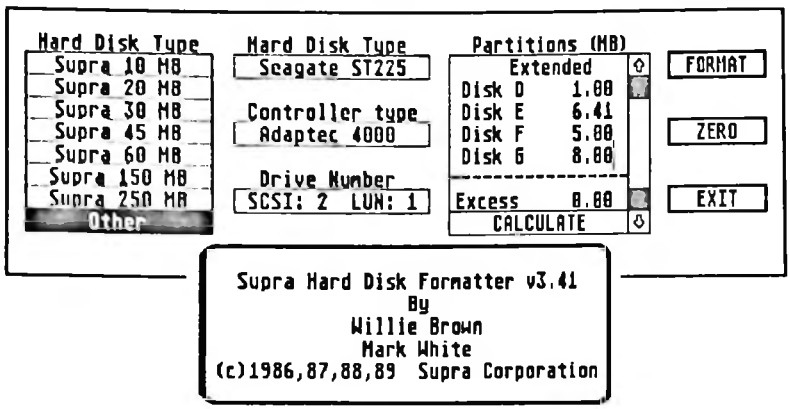


Figure 7.4

MISCELLANEOUS ITEMS

1. You should be able to connect any drive to your ST that has an ST506/ST412 or SCSI interface, regardless of whether it's a 3.5-inch, 5.25-inch, half height or full height drive. (A full height drive will take up twice as much space as a half height drive, so you'll have to plan accordingly.)

2. In deciding what brand and size of drive, and what controller to buy, contact the technical support staff at the company that manufactures your host adapter. They can tell you which drives are reliable and which drive/controller combinations will work.

3. You may save yourself some time if you try out your components before you install them into the case. Put the host adapter, controller, drive, etc., on a table, connect all the cables (including the cable to the ST), then see if you can properly format the drive. If you can't, and you want to pull out cables or move components around, it will be a lot easier than if the components are already bolted into the case.

4. You can buy the necessary ribbon cables from your host adapter's manufacturer, and probably from the dealers who sell drive mechanisms and controllers. To build your own cables, check the ads in *Computer Shopper Magazine* or *Byte Magazine* for supplies. I buy ribbon cable and connectors from JDR Microdevices, 2233 Branham Lane, San Jose, CA 95124. Ph: (800) 538-5000, or (408) 559-1200.

5. Before crimping connectors to ribbon cables, you might want to examine the contacts on the connectors to make sure they're not damaged. I didn't do this, and when I then had trouble pushing a connector onto the edge card of a drive, I discovered that one of the contacts on the connector was bent. Unfortunately, it damaged the edge card on the drive.

6. Don't buy too small a power supply. Drive mechanisms can draw between 11 and 25 watts each, and can also draw between 3 to 4 times their operating current when they're starting up. The more drives, controllers, etc., that you add, the more power you'll need. A 150-watt power supply should work fine.

7. Even if your power supply has a fan, it will probably draw

out only the heat created by the power supply itself. You'll want to install an additional fan to draw out the heat generated by your drives. I bought a fan from Radio Shack and positioned it near the back of my XT case, after removing some of the port covers.

8. I have purchassed components from the following suppliers:

- Hard drives: Hard Drives International, 1912 W. Fourth St., Tempe, AZ 85281. Ph: 800-234-3475 or 602-967-7435.

- Controllers: Computer Surplus Store, 715 Sycamore Dr., Milpitas, CA 95035. Ph: 408-434-0168.

Chapter Eight

CONNECTING 5 1/4 - INCH FLOPPY DRIVES

QUESTION: I'd like to bring home some 5 1/4 - inch floppy disks from an IBM computer at work and use them in my ST. Can this be done?

ANSWER: Yes, but in order to do so you'll need to connect an IBM-type 5 1/4 - inch floppy disk drive to your ST. (Once the drive is connected, you'll also be able to save files that are in ST-format onto 5 1/4 - inch floppies.)

When I first contemplated hooking up a 5 1/4 - inch drive to my ST, I expected to encounter a lot of problems, but the project turned out to be pretty easy. Here's what's involved:

The first step is to decide whether to buy an already-assembled drive or to purchase the individual parts and assemble your own drive system. If you're not the mechanical type, you'll probably want to choose the first option. Five and one fourth inch drive systems that are ready to plug into the ST are available from IB Computers (9244 S.W. Beaverton-Hillsdale Hwy., Beaverton, OR 97005. Ph. 503-297-8425) or E. Arthur Brown Co. (3404 Pawnee Dr., Alexandria, MN 56308. Ph. 612-762-8847). Atari has also announced a 5 1/4 - inch drive for the ST (model # PC554).

There were two reasons why I chose to assemble my own drive system. First, I thought it would be cheaper, and second, I felt I

would learn more by doing it myself. I definitely learned more, but I probably spent almost as much money as I would have had I bought one of the assembled units.

Assembling your own drive system involves some major dangers. Before even *considering* this project, read the cautionary words and warnings that appear in Chapter 7 (See page 110).

Assembling your own system involves the following steps:

1. Deciding what type of drive to purchase (and making any necessary modifications to the drive).
2. Buying or making a cable to connect the drive to the ST.
3. Putting the drive in a case.
4. Connecting the drive to a source of power.
5. Running step-rate-setting software that allows your ST to properly interact with the new drive.
6. If you want to use your ST to format 5 1/4 - inch floppies that can be used on an IBM computer, you'll also need special formatting software.

DECIDING ON A DRIVE

Not all drives that work on an IBM computer are the same. In fact, a number of different manufacturers make drives for the IBM and IBM clones, and the specifications of these drives can vary. ST users have been successful in hooking up a number of different IBM-style drives, but some installations are more difficult than others.

However, there is at least one 5 1/4 - inch drive that can be connected to the ST without the need for any modifications. This is the TEAC 55BR, Series 100, which I was able to purchase new for $79 from J.B. Technologies (21011 Itasca St., # F, Chatsworth, CA 91311. Ph. 818-709-6400). (TEAC 55BR drives with series numbers

other than in the 100s may require de-soldering or clipping out of resistors, as mentioned below. Therefore, for the easiest installation, make sure that you're getting a series 100 unit. My drive says "TEAC FD-55BR 102U" on it.)

The 55BR is a 40 track double density drive. Depending on what kind of drives are in the IBM computer that you want to share floppies with, you may want to purchase for your ST something other than a 40 track double density drive.

The following are some of the 5 1/4 - inch floppy disk formats that have been used in IBMs and clones:

180 k (40 track, single sided)
360 k (40 track, double sided)
360 k (80 track, single sided)
720 k (80 track, double sided)
1.2 m (80 track, double sided, high density)

The following 3 1/2 - inch floppy disk formats have been used in IBMs and clones:

360 k (80 track, single sided)
720 k (80 track, double sided)
1.44 m (80 track, double sided, high density)

For information about connecting IBM-style drives to the ST, you might want to buy a booklet called, *ST 5.25 Drive & Emulation Manual*, published by E. A. Brown Co (see address above). This 19-page pamphlet discusses how to attach the following drives to the ST:

TEAC 55BV
TEAC 55BR
TEAC 55F

TEAC 55GFR
Epson SD-521L
Tandon TM100-2 and TM100-4

You may also want to use your modem to visit GEnie's Atari ST RoundTable. In category 4 (hardware), topic 13, I found 103 messages (as of March 1989) discussing the use of 5 1/4 - inch drives on the ST. Some of these messages discussed the connection of the following additional drives to the ST:

Shugart 455
Remex RD-480
Panasonic JU475
Panasonic JU-455-5
Tandon 65-21
Qumetrack 142

However, in order to use some of the 12 drives listed above, "pull-up resistors" may have to be removed from the drives. These resistors are involved in the computer's ability to recognize the drive. With one or two of the above drives, the resistors can be easily popped out with a screwdriver. On some of the other drives, however, the pull-up resistors are soldered to the drive board. On these boards, you may have to de-solder the resistors or clip them out with small wire cutters. You may want to try using a drive first before fooling with its resistor packs. If the drive doesn't work, then you can worry about the pull-ups.

BUYING OR MAKING A CABLE

You can construct your own drive cable, or buy an already-made one. I bought an assembled cable, since I didn't want to risk assembling a home-made cable incorrectly. Making your own cable is a fairly dangerous project. If you connect even one wire

improperly, you could damage your ST or your drive.

If you decide to make your own cable, you'll need a 14-pin DIN connector to plug into the drive port on the back of your ST, as well as a 34-pin edge connector, which will connect to the 5 1/4 - inch drive.

Thirty-four pin edge connectors are easy to find. Many advertisers in *Byte Magazine* and *Computer Shopper Magazine* sell them, as do Radio Shack stores. Fourteen pin DIN connectors are somewhat harder to find.

The easiest way to acquire the 14-pin connector is to buy an extra Atari ST disk drive cable and chop off one of its ends. However, there are at least two sources for the 14-pin DIN connectors: IB Computers (see address above), and Best Electronics (2021 The Alameda, #290, San Jose, CA 95126. Ph. 408-243-6950).

MY SUGGESTION: BUY THE CABLE

A number of public domain files contain diagrams describing which pins on the 34-pin connector should be attached to which pins on the 14-pin DIN connector. For example, ST—IBMDD.PI3 (a public domain file) consists of a Degas picture showing the pin connections; and IBMDISK.DOC (also in the public domain) is a short text file discussing pin connections and sources of DIN plugs. Information about pin connections is also contained in the E. A. Brown Company booklet that comes with their cable. However, the instructions in these three sources don't agree on which pins from the ST's floppy port should be connected to which pins on the drive. Because of this, I strongly recommend that you spend the extra $20 to buy a cable from IB Computers or E. Arthur Brown Co.

Regardless of whether you buy a pre-assembled cable or construct your own, plugging in the 5 1/4 - inch drive is easy. Refer to your ST owners' manual and follow the instructions that describe how to connect a second floppy drive. If you're already using two 3 1/2 - inch drives, you'll have to disconnect the second drive before connecting the 5 1/4 - inch drive, since the ST can only

accommodate two drives. (A number of different companies make switch boxes specifically for the ST that allow you to switch between two drives. You'll then be able to have three drives connected to your ST. Ask your Atari dealer about such switches.)

The 34-pin edge connector on the cable gets pushed onto the 34-pin edge card on the back of the drive. However, you have to be very careful that pin "1" on the cable's connector gets lined up with pin "1" on the drive's edge card. Refer to Chapter 7 for instructions on how to identify pin 1 on a ribbon cable (See page 115).

WARNING: Turn off your computer and your drive before connecting or disconnecting drive cables.

SOFTWARE

An Atari ST sends its drives head step commands at a rate of 3 ms (milliseconds) per track. Most IBM-style 5 1/4 - inch drives, however, aren't capable of going that fast. Luckily, the ST's drive controller chip can be software selected to handle drives with alternate head stepper times.

Before you can use a 5 1/4 - inch IBM-style drive, you'll probably need to run a program that sets the ST's head stepper time to match that of your 5 1/4 - inch drive. Here's a list of a few of these programs:

1. PCF554.PRG. Created by Atari Corporation, this public domain program works on TOS 1.0, 1.2 and 1.4.
2. SIXMS.PRG. (Public domain.)
3. STEPPER.PRG, by E. A. Brown Company. (See Figure 8.1) This is a commercial program (i.e., not public domain) and is provided with the 5 1/4 - inch cable the company sells. One nice feature of this program is its ability to disable an altered step rate (and return it to 3 ms) without having to reboot.

PUTTING YOUR DRIVE IN A CASE

A floppy drive contains a number of delicate parts, so you don't want your new 5 1/4 - inch drive just lying on your desk. (Of

course, if you've bought a pre-assembled system, the drive will already be in a case.)

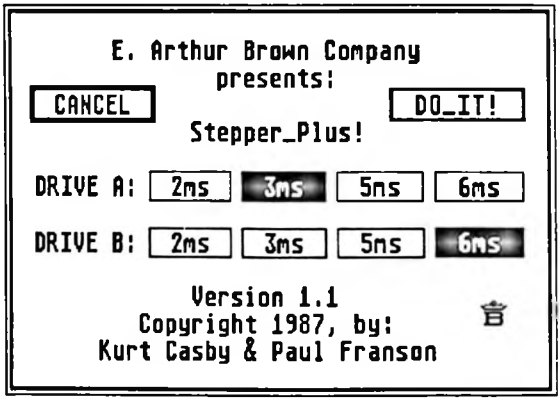


Figure 8.1

You can buy a case that's specifically designed to house a 5 ¼-inch drive. Check the ads in Computer Shopper Magazine. If you already have a hard drive, check to see if its case has room for an extra drive.

POWER

Drives from IB Computer or E. A. Brown Company come with their own power supplies. If you put together your own system, however, you'll have to supply a source of power. Most IBM-style drives have a standard receptacle which matches up with the leads coming from the power supplies designed for IBM computers. The drives typically use +5 and +12 volts. I purchased a 150-watt power supply that has 6 sets of leads. One set goes to my 5 ¼-inch floppy drive, and other sets go to the hard disk and host adapter. I can use the remaining sets to add additional drives or components in the future.

FORMATTING YOUR NEW DRIVE

There are many messages on GEnie and Compuserve concerning the formatting of 5 1/4 - inch floppies. Some comments warn against using the ST's Desktop formatting routine (the one you're most familiar with) to format a disk in a 5 1/4 - inch drive. Other messages merely say that if you format from the Desktop, the formatted disk won't be readable when used in an IBM computer.

Instead of using the Desktop's formatting routine, you can run one of the many formatting programs for the ST that create a true MS-DOS/PC-DOS formatted disk. Here are some examples of such programs:

1. IB Computer's ST—FORMAT.PRG (See Figure 8.2) This is a commercial program (i.e., not public domain) and is provided with the 5 1/4 - inch cable, and the drives, that the company sells.

```
                    ST FORMAT  VERSION 1.02

            © 1986, BY DAVID SKINNER, AURORA OREGON


 0 SINGLE OR DOUBLE SIDED FORMAT S/D

 0 DESTINATION DRIVE? A/B

40 NUMBER OF TRACKS 0-83

 9 SECTORS PER TRACK 8-10

 0 CODE FOR STEP RATE SPEED 0-3
   0=SLOWEST/3=FASTEST


                          PRESS <UNDO> TO EXIT
```

Figure 8.2

2. MULTIFMT.PRG, by E. A. Brown Company (See Figure 8.3) Just like the program from IB Computer, this is a commercial program (i.e., not public domain) and is provided with the 5 $^{1}/_{4}$ - inch cable, and the drives, that E. A. Brown sells.

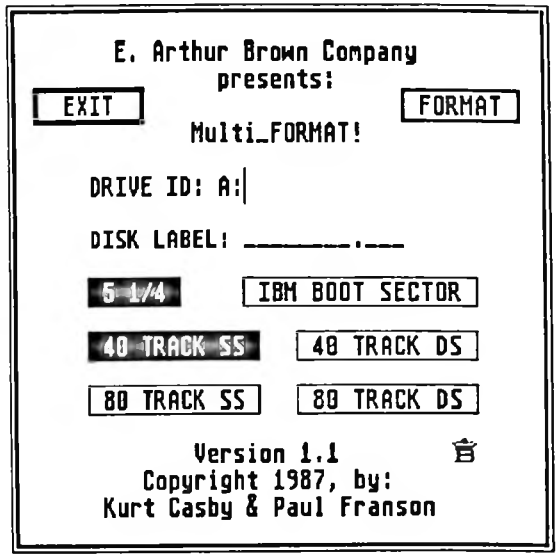


Figure 8.3

3. FORMAT40.TOS (Public domain.)

Chapter Nine

IBM EMULATION

Although the Atari ST is superior in many ways to a comparatively priced IBM computer, there nevertheless are a number of valid reasons for wanting to run IBM programs on your ST.

First, since there are many more IBM personal computers than there are STs, a greater amount of software is available for the IBM than for the ST. And although much of the software for the ST is excellent (and sometimes even superior to what's available for the IBM) there are certain types of software that are available only for IBM computers. Second, if you use an IBM at work, you may want to bring home your IBM disks and continue the work on your ST.

At the time this book went to press, there were four IBM emulators for the ST that were being shipped (or had been announced and which were expected to be released soon). Three of these were hardware emulators, one was a software emulator.

Both hardware and software emulators accomplish the same thing: allow you to run IBM software. A hardware emulator does this with hardware, and has its own CPU and other chips. A software emulator is a program that is supplied on a regular ST floppy disk. You run the program on your ST and the ST is fooled into thinking that it's an IBM computer.

The retail prices of the three hardware emulators range from $299 to $399. (Note: much of the following information about these products is based solely on telephone conversations I've had with the

products' suppliers. In other words, I have not been able to personally verify all of the stated specifications. Therefore, before you make a decision about buying an emulator, you'll want to do some investigating of your own.)

PC SPEED

(MichTron, 576 So. Telegraph, Pontiac, MI 48053. Ph: 313-334-5700.) (Micro-D, 391 Ormont Dr., Weston, ON M9L 1N8 Canada. Ph: 416-741-9825) (Paragon Computers, 5 Short St. # 17A, Perth, W. Australia 6000. Ph: 09 221 3216) (Gasteiner Technologies Ltd. 8 Thornaby Gardens, Edmonton, London, Great Britain. N18 Ph: 01 884 2415.)

1. Norton SI rating of 4.0. (The higher the Norton rating, the better. IBM programs will run faster with an emulator that has a high Norton rating than with an emulator with a low rating.)
2. Uses the NEC V30 8 Mhz microprocessor.
3. Installation requires opening the ST's case and soldering a socket onto the top of the ST's 68000 microprocessor. Unless one has sufficient experience with soldering, MichTron recommends that the installation be done by an Atari dealer.
4. Uses your ST's memory. On a 520ST, 256k of memory is available; on a 1040ST or Mega there's 704k available.

SUPERCHARGER

(Talon Technologies, 243 N. Highway 101, # 11, Solana Beach, CA 92075. Ph: 619-792-6511.) (Paragon Computers, 5 Short St. # 17A, Perth, W. Australia 6000. Ph: 09 221 3216) (Condor Computer, Great Britain. Ph: 01 828 9755.)

1. This is a stand-alone unit that plugs into the ST's DMA port. It has a through port that allows use of the Atari laser printer, hard drives, etc.
2. Norton SI rating: ? (I've seen reports of 4.4, 4.0 and 3.0.)
3. Uses the 8 Mhz V30 microprocessor.

4. Comes with its own memory: 512k is supplied, and there are sockets for adding an additional 512k.

5. When in PC mode, you can use your ST's memory as a RAM disk. Likewise, when in the Atari ST mode, you can use the SUPERCHARGER's memory as a RAM disk.

6. SUPERCHARGER has a bus connector for use with peripheral devices. An additional box, which will plug into the bus and accommodate standard IBM cards, is being planned.

7. An empty socket allows addition of an 8087 math co-processor.

8. "Hot keys" permit you to switch back and forth between the PC and Atari ST modes. When in PC mode, if you press [Shift]+[Alt]+[Backspace], you'll shift into the Atari ST mode. If you hit these keys again, you'll find yourself exactly where you left off in PC mode.

9. Supplied with Microsoft DOS 4.0.

pc-ditto I

(Avant-Garde Systems, 381 Pablo Point Dr., Jacksonville, FL 32225. Ph: 904-221-2904) (Micro-D, 391 Ormont Dr., Weston, ON M9L 1N8 Canada. Ph: 416-741-9825) pc-ditto I is a software emulator which uses your ST to imitate an IBM PC XT computer. To use the program, you need:

1. The pc-ditto disk.

2. An Atari ST.

3. A copy of the IBM operating system, either IBM's PC-DOS, Microsoft's MS–DOS, or a DOS clone.

4. IBM software.

Not all IBM programs can be run using pc-ditto I, but the list of programs that do run is impressive: Lotus 1-2-3, Q&A, Sidekick, Norton Utilities, Flight Simulator, PC-File, dBase II, III, III Plus, GW Basic, Word Perfect, Wordstar and hundreds more. You can contact Avant-Garde to receive a longer list of programs that will run.

The major disadvantage of pc-ditto I is slow speed. This software emulator runs IBM programs at approximately one third the speed of an IBM PC XT computer.

pc-ditto II

(Announced. Also from Avant-Garde Systems.)

1. Basically the same product as pc-ditto I, except much faster. Has a Norton SI rating of 3.0.
2. 10-Mhz CPU.
3. Installation requires opening the ST's case and seating a socket onto the top of the ST's 68000 microprocessor.
4. Uses your ST's memory. On a 520ST, 384k of memory is available; on a 1040ST or Mega there's 640k available.

USING A 5 1/4- INCH FLOPPY DRIVE WITH AN EMULATOR

The earlier IBM PCs had 5 1/4 inch floppy disk drives, but newer IBM models come with 3 1/2 inch drives. Since many of the IBM programs, as well as DOS, are now available in both formats, you don't need a 5 1/4 inch drive connected to your ST in order to use one of the IBM emulators. That is, you can use an emulator with nothing more than your ST's single-sided or double-sided floppy disk drive.

However, it's quite easy to connect an IBM-style 5 1/4 inch floppy disk drive to your ST. You can purchase a pre-assembled drive, or construct your own system. (See Chapter 8). Why would you want a 5 1/4 inch drive? In a number of situations, a 5 1/4 inch drive may be necessary, or at least very useful:

1. There are a number of IBM programs which are available only on 5 1/4 inch disks. Although a non-copy-protected program that's on a 5 1/4 inch disk can be copied onto a 3 1/2 inch disk, this procedure won't work with copy-protected software. Nor will one of the so-called copy protection busting

programs help: although you'll be able to copy the master disk, the copy you end up with will still contain the copy protection, which will prevent the program from being successfully copied onto a 3 1/2 inch disk. If, however, your ST is equipped with a 5 1/4 inch drive, you'll be able to run even copy-protected IBM software on 5 1/4 inch disks.

2. If your only copy of DOS is on 5 1/4 inch disk, you'll have to do one of two things: (1) connect a 5 1/4 inch drive to your ST, or (2) copy DOS from the 5 1/4 inch disk onto a 3 1/2 inch disk.

Although connecting a 5 1/4 inch drive is obviously the more expensive of the two alternatives, by doing so you'll end up with a system that is very convenient and versatile. If, however, you'd rather just copy DOS, find someone with an IBM computer that has both a 5 1/4 and a 3 1/2 inch drive (or someone with an ST that is equipped with a 5 1/4 drive, a 3 1/2 inch double-sided drive, and and an IBM emulator. It's not possible to perform the copy operation on an ST that has only a single-sided drive). If you take your 5 1/4 inch DOS disk to an IBM dealer, he should be able to perform the copy operation for you. In either case, the copying should be done from within DOS, using the FORMAT command.

Chapter Ten

MISCELLANEOUS SUBJECTS

GARBAGE CHARACTERS IN DIRECTORY

QUESTION: I received a public domain disk in the mail from an ST magazine. But when I attempted to de-arc several of the programs, it was apparent that something was wrong with the disk. You can see from the enclosed screen dump of the disk's window that there are a lot of garbage characters in the file names. (See Figure 10.1)

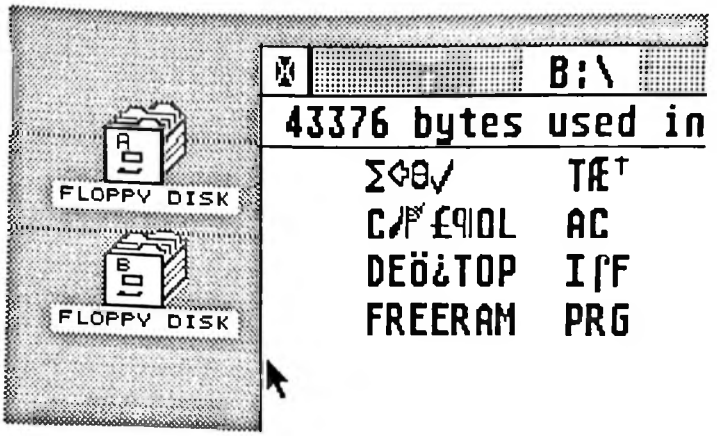


Figure 10.1

When I tried to load a program from the disk, it didn't work. And after making a copy of the disk, the copy wouldn't respond to the disk drive! I have no problem loading and using other programs and disks. I worried that maybe there was a virus on the disk, but when I checked it using the virus detecting program, VKILLER.PRG, by George Woodside, the disk checked out OK.

ANSWER: I, too, can find no indication that there's a virus on your disk. Instead, what the disk most likely is exhibiting is a deterioration of the surface of the circular floppy disk that spins around inside the floppy's $3\,^1/_2$-inch case.

As was described in Chapter 5, the heads in our disk drives magnetize the surface of floppy disks in a pattern that corresponds to the binary values of the information that's been sent to the disk. If one small area of the disk is magnetized in one direction, the binary digit "0" is stored. If, instead, that area is magnetized in the opposite direction, the binary digit "1" is stored.

It takes eight binary digits (i.e., eight 0's and 1's) to represent a letter or number, such as the letter "H." If even just one of those eight bits is changed, there no longer will be an "H" stored at that point on the disk. Depending on which bit(s) are changed, the "H" may be turned into another letter of the alphabet, or a number, or a "strange character," as in your case.

The surface of a disk can become defective and incapable of reliably storing the proper magnetic patterns. Over time, a disk that was once good can become bad. It appears that your disk was defective right from the start.

If you ever notice that an important disk's directory is corrupted, immediately make a copy of the disk (if you can). If the deterioration on the original disk continues, you may eventually be unable to get a drive to even recognize the disk. Sometimes, though, you'll be able to salvage some files from the copy, provided they were on a part of the original disk that wasn't defective.

The fact that disks can deteriorate is a major reason why you should maintain backups of important disks. Any one of your disks can go bad at any time, and sometimes this can happen abruptly. The first indication that something is wrong could be a warning message that data on the disk may be bad, and at that point it may be too late to make a backup.

PRINTING SERVICE BUREAUS

QUESTION: The quality of the printing from my dot matrix printer is not good enough for certain things, such as business presentations. Are there any places where I can pay to have files printed out on a laser printer?

ANSWER: Yes. Printing service bureaus exist that specialize in creating high quality printouts on a per-page basis for Atari ST owners. To find out whether there's one of these businesses near you, call your local Atari dealer or talk to members the closest ST users' group.

But even if there's not a service bureau in your area, you can make use of a bureau in another town, or even in another part of the country. A growing number of ST owners use the postal service (or United Parcel Service, etc.) to send a file on disk to one of these printing shops. Some service bureaus even accept files via modem. Then, once the printing is complete, the hard copy is returned to you in the mail.

To find out what the price for this kind of printing is, I called a number of Atari printing service bureaus. Maxwell Graphics and Publications (507 West Baseline, Lafayette, CO, 80026, phone 303-665-4849) charges $2 per page for one to ten pages, or $1.50 for ten to twenty pages. They accept files that have been created in the following programs: Pagestream, Calamus, Timework's Desktop Publisher and Easy Draw. Additionally, they can print files in Postscript format from Pagestream or Timework's DTP.

Maxwell Graphics can also take a file that's been created on your ST and produce a very high quality output on their Linotype machine. The cost is $10 per 8 $^1/_2$ x 11 - inch page, and you have your choice of 1270 or 2450 dpi (dots per inch). (Laser printer output is usually 300 dpi; 9-pin dot matrix copy is often around 120 dpi; and 24-pin dot matrix copy can be somewhere between 180 and 360 dpi. However, many people feel that even a printout from a 360 dpi 24-pin dot matrix printer doesn't look as good as output from a 300 laser printer. This is because a dot matrix printer's ribbon limits the degree of resolution that can be obtained.) Maxwell Graphics, which is also an offset printshop, has a 24-hour modem (303-666-4470) so it's possible to get a file to them very quickly by

uploading it from your ST.

Compuplace Inc. (3585 Keele Street, #12, Downsview, Ontario, M3J 1M7 CANADA, Ph: 316-633-8667) operates a similar service. They accept files from Pagestream (including files in Postscript format), Calamus, Timework's Desktop Publisher, WordPerfect, Word Writer and 1st Word Plus. The charge is $1 per page for single pages, with discounts for larger runs. Compuplace is also an Atari dealer.

Incidentally, there are a number of other things that a disatisfied dot matrix printer owner should consider. First, make sure you're actually getting the best quality from your existing printer. For instance, although most printer owners know how to shift out of the poorer quality draft mode and take advantage of the "near letter quality" mode, many dot matrix printers also have a "double printing" and/or an "emphasis" mode. These features may produce output that's even better looking than near letter quality. Read your printer's manual to see whether these modes can be button-selected by pressing switches on the printer, or whether they have to be accessed with printer control codes from within your software.

Finally, consider renting a laser printer, or buying a printer like Hewlett Packard's DeskJet Plus, which produces inkjet copy at 300 dots per inch.

CAN'T DELETE FOLDER

QUESTION: After the time-consuming task of de-arching 70 files of LODRUNR2 from a public domain disk, the program wouldn't load, due to a missing music file that would not de-arch.

Wanting to delete the worthless LODRUNR2 files from my DE-ARC disk, I tried to drag the folder in which they were located to the Trash Can, but a puzzling dialog box appeared. (See Figure 10.2)

So then I started dragging individual files to the Trash, which worked until I came to the HIGH.SCR file. That is the only remaining file in the LODRUN2 folder and I can't get rid of it even after a cold reboot. Please tell me what's happening.

ANSWER: There's a simple answer to why your ST won't allow you to trash that file. But after we solve this problem, I want to digress by discussing a variation of your situation which many people may not be aware of.

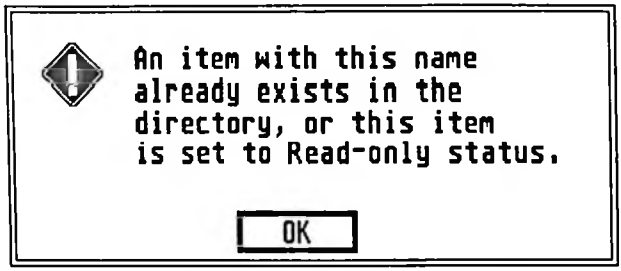


Figure 10.2

You weren't able to trash the HIGH.SCR file because the file's attribute had been set to "Read-Only." When a file is set to Read-Only, it becomes impossible to change, rename, or delete the file.

If a file has been set to Read-Only and you want to change, rename, or delete it (as you wanted to do), you first have to change its attribute to "Read/Write." Read/Write is the opposite of Read-Only, and is the normal, default status for a file. When a file is set to Read/Write, it can be modified or deleted.

To change your file's status to Read/Write, go to the Desktop and mouse-click once on the file's icon (or its name, if your files are displayed as text). Next, choose Show Info from the FILE menu, and when the "Item Information" box appears, use the mouse to click on the Read/Write button. Finally, click on "OK," and you should now be able to delete the file.

Not only is it impossible to delete a Read-Only file, but if that file resides inside a folder, then you won't be able to delete the folder. This is true even if the folder contains other files which have not been set to Read-Only, as you discovered.

Read-Only is to individual files what "write-protect" is to floppy disks. (When the write-protect notch on a floppy disk is closed, you won't be able to write or delete anything from the disk, or format the disk. The notch is considered closed when you can't see

through the square hole.)

If you want to protect all the files on a disk, the write-protect notch is the easiest way. Closing the notch is also a good way to protect a disk from a virus. Once the disk is write-protected, nothing can be written to the disk, including virus code.

However, there will be times when you'd like one or more (but not all) files on a disk to be protected from alteration. In these cases, write-protecting the disk is inappropriate, since doing so will prevent you from doing anything other than loading files from the disk. You won't be able to save or update any of the files on the disk.

Contrasted with this, the Read-Only option gives you a lot of flexibility. For example, some users set the status of their DESKTOP.INF file to Read-Only. They've found that it takes quite a bit of time to set up the Desktop the way they want it, and if the file's status is Read/Write, an inadvertent saving of the Desktop will write over their carefully created DESKTOP.INF file. (This might or might not be a problem, depending on how the Desktop is arranged at the time the Desktop is inadvertently saved.)

The Read-Only attribute can also be useful where more than one person uses the computer. If you have an important data file on a disk, and that disk must be used by other people, you can limit the changes of the file being altered or deleted if you change its status to Read-Only.

Now for the digression I promised. I discovered an interesting anomaly while experimenting with files whose attributes have been changed to Read-Only. After making a copy of a file which had been set to Read-Only, I found that the copy's attribute was Read/*Write*. In other words, the original file's Read-Only protection had not been retained during the copy operation.

If a user is unaware of this, and he consequently relies on the copy of a file being Read-Only protected, data could conceivably be lost. Note, however, that although this bug exists in the version of TOS that came with my 1040ST, the situation is expected to be corrected in future versions of TOS.

TRASH CAN PROBLEM

QUESTION: When I try to drag a folder or file icon to the Trash icon, why do I frequently get a dialog box saying, "You cannot drag folders," etc. "onto the Desktop"? I have had to drag the same file to trash as many as seven times before the Delete box would appear and permit the option to work.

ANSWER: A number of things can cause this. First, once you've selected the item to be trashed, and while it's being dragged to the Trash can, it's possible to inadvertenly let up momentarily with the left mouse button. This release of the button will cause the dialog box to appear.

However, it sounds like your trouble is occurring after you've reached the Trash can. Once you're over the Trash can, the deletion will be successful only if the mouse arrow is positioned directly on top of the Trash can at the moment you let up on it.

This sounds simple, and usually is. However, in order to understand your problem, a distinction must be made between (a) positioning the *mouse arrow* over the Trash can, and (b) positioning the *item to be trashed*, over the Trash can. The former is what you want to do, and by doing that, the latter may, or may not, also occur at the same time.

To understand this, let's do some experimenting. From the Desktop, mouse-select a file from a window, then drag it over the Trash can. If you move the file around a bit without letting up on the left mouse button, you'll notice that at some point the color of the Trash can will change. When this change occurs, stop moving, and let up on the mouse button. The color change indicated that you were properly positioned over the Trash can.

Now look at Figure 10.3, which shows an incorrectly positioned file. As you can see, although the file is positioned directly over the Trash can, the mouse arrow isn't. If the mouse button is released in this situation, the alert box you've described will appear.

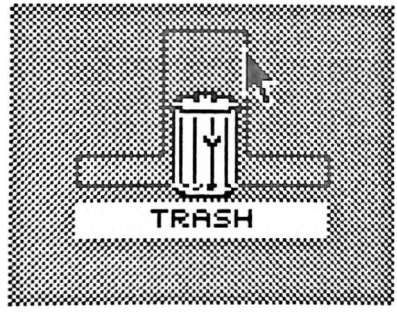


Figure 10.3

HOW TO PEP UP A SLUGGISH MOUSE

There's something else that may be causing or contributing to your problem. It's possible that your mouse isn't functioning as well as it should. If mouse movements in addition to trashing don't seem as easy as they used to be (e.g., they are jerky or sporadic, or the mouse arrow seems to move slowly), it's time to give your mouse a checkup.

First, make sure that your computer is turned off. Next, perform the mouse-ball cleaning instructions that appear in your computer owner's manual or your mouse manual. Once the mouse ball is removed, look into the ball cavity and observe the three rollers which allow the mouse to rest on the ball. You may find that these rollers, or the axles on which the rollers spin, have become clogged or caked with what can euphemistically be called "foreign matter."

This foreign matter is probably a combination of house dust, sweat, and oil from your skin. Sweat and oil that have rubbed off of your hand can accumulate on the mouse pad, and when they do, house dust will stick to this sweat/oil accumulation. After building up on the top of the mouse pad, this junk gets onto your mouse ball and may collect on or around the mouse rollers.

Such a buildup of foreign matter can greatly diminish the performance of your mouse. I'll describe how I remove this stuff, but let me also tell you that, although this procedure is very simple, I perform it very carefully, since I don't want to damage my mouse.

(Mice are expensive to replace.)

I use two items to perform the cleaning: a pair of tweezers, and a plastic cover from a ball-point pen. Although I use the kind of plastic cover that has a narrow one-inch protrusion which allows the pen to be clipped to a pocket, other thin pieces of plastic might work, provided that they would allow me to easily and safely remove the junk from the rollers.

The reason I mention safety, and the reason that I use a plastic tool is because I don't want to scratch the surface of the rollers, or their axles. (The rollers in my mouse are metal. But even those mice with plastic rollers should be treated carefully.) I gently scrape the junk off of the rollers with the plastic, then lift it out with the tweezers.

If the buildup of junk inside the mouse isn't very large, I forego the tweezer/plastic method and merely use a cotton swab. However, I'm careful not to use any solvent, water, or any other liquid which would rust or otherwise damage any of the mouse's parts. In fact, it's best not to get liquid of any kind even near the mouse, or any other part of the computer. Liquids can damage your computer's (and your mouse's) printed circuits, and other parts quicker than you can say, "Help! Where's the nearest Atari repair shop?"

Here's a piece of preventive advice. Cover your mouse pad when you're not using your computer. If left uncovered, dust will accumulate on the pad and will eventually find its way into your mouse.

One final observation. You are using a mouse-pad, aren't you? If not, you might want to buy one. Not only does a pad make mousing around more pleasurable, but it probably will extend the life of the mouse ball. I suspect that the surface of a mouse ball that's being rolled over a hard table will lose its smoothness more quickly than a ball that's being rolled over a soft sponge rubber pad.

BACKUP FAILURE

QUESTION: For important data base records such as my customer list, I maintain two copies of a file, with each copy on its own separate disk. After finishing a session with the data base

program, I return to the Desktop and drag the icon for my master file onto the icon of the disk containing the backup copy of the file.

This system worked fine for many months. Then one day when I tried to load the master file into the data base program, I got an error message saying that the file couldn't be opened.

Although this surprised me, I didn't worry, since I had the backup copy. However, when I tried to load the backup file, the same error message appeared! Even though other files on each disk were readable, I couldn't get the program to load either copy of my customer list.

A friend looked at the files and said that both of them were corrupted. After a lot of work, he somehow was able to recover my file. But this whole thing really scared me. How could both my master file and its backup get corrupted on the same day?

ANSWER: Here's what may have happened. During the data base session just prior to when you received the error messages, the master copy of your file somehow became corrupted, possibly while the file was being saved. The corruption could have been caused by your data base program, the ST's operating system, or by a physical abnormality on the disk. Whatever the cause, your master file was no longer usable. However, at this point the backup copy was still uncorrupted.

When you returned to the Desktop and copied the master file onto the backup disk, this erased the old backup file. You ended up with a corrupt master file and a copy of that corrupt file on your backup disk.

Although your backup system didn't work in *this* instance, it was better than no system at all. It provided protection against certain types of losses, such as those caused by inadvertantly dragging one copy of a file to the Trash can.

However, in order to avoid the sort of problem you experienced, you need a more sophisticated backup routine. I sometimes employ a system that uses three disks. I call my system a "staggered backup procedure," and here's how it works:

I save three copies of any important file, and each file is on its own separate disk. I call the three disks:

"Master disk"
"Backup disk # 1"
"Backup disk # 2"

It doesn't matter whether these disks are hard drives or floppy disks. However, each one should be a separate disk. For instance, if you don't have a hard disk, use three separate floppy disks. If you have a hard drive, the hard drive can be the master disk; backup disk #1 can be a floppy; and backup disk #2 can be another floppy. (Note: if you have only one hard disk, use two floppies for the backup. In other words, it's not sufficient to put the master on one partition and the backups on other partitions of the same hard drive.)

The first time you use a program (a data base manager, a word processor, etc.) to save a new file, perform the following two steps:

- Save the file from within the program onto the Master disk.
- Go to the Desktop and drag the file's icon from the Master disk onto Backup disk #1.

From then on, after each use of the program, do the following:

1. Save the file from within the program onto the Master disk. (See Figure 10.4)
2. Go to the Desktop and drag the file's icon from Backup disk #1 onto Backup disk #2.
3. Drag the file's icon from the Master disk onto Backup disk #1.

If you use floppies, make sure that the names of all the floppies are clearly marked on the covers of the disks ("Master disk," "Backup disk #1," etc.). This is important because, if you perform one of the copy operations onto the improper disk, the procedure won't provide maximum protection. It's also vitally important to perform the steps exactly in the order listed.

By using this procedure, backup disk #2 will always contain a copy of your file that is one session *older* than the copies on both the master disk and backup disk #1. Here's how this fact will limit

the chances of your type of disaster from happening again.

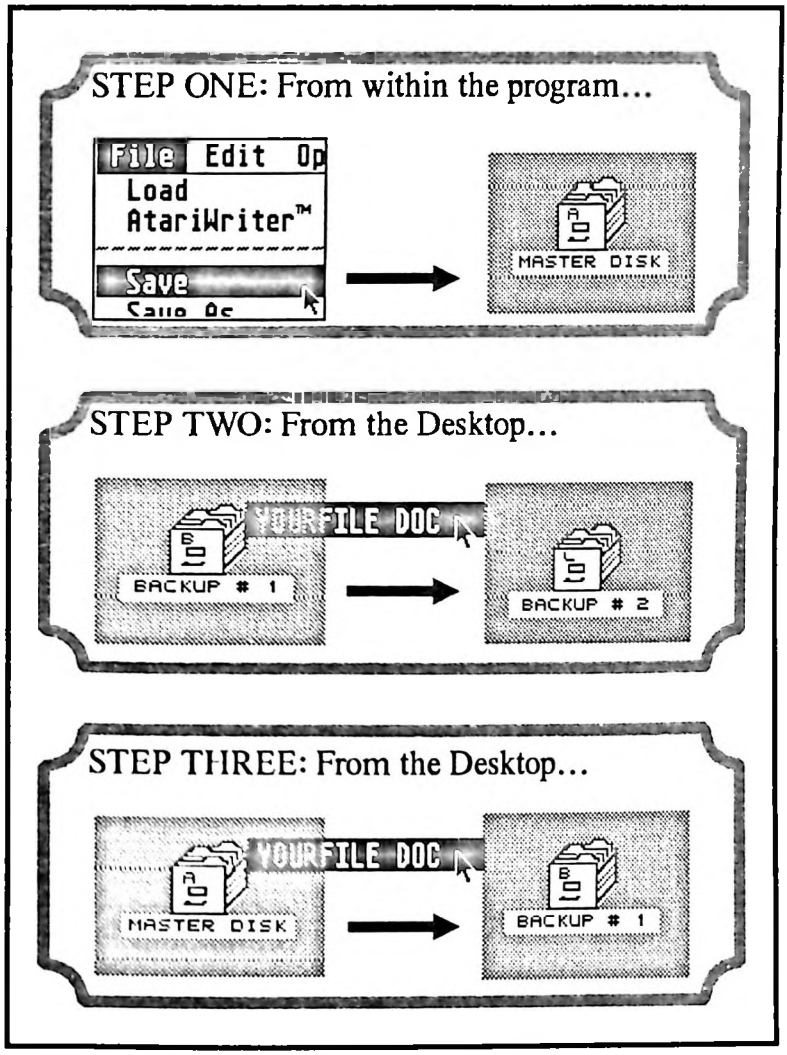


Figure 10.4

If an error occurs during the saving of a file to your master disk,

you will learn of the error *before* you alter the file on backup disk #2. You'll still get the error message saying that the program can't load the master file, but you still should be able to load the file from backup disk #2.

Although you won't be able to recover any new data that you entered during the last session (since backup disk #2 doesn't contain that day's file), you *will* recover the file from the session just prior to the last session.

If you'd like even more protection than this three-disk system provides, there are other more sophisticated routines that you can follow. For instance, with a seven-disk staggered system there would be one disk for each of the five business days of the week:

Master disk
Backup disk #1
Monday's Backup disk
Tuesday's Backup disk
Wednesday's Backup disk
Thursday's Backup disk
Friday's Backup disk

Here's how this system would work: The first time you use a data base manager or word processor to create a new file, perform the following two steps:

- Save the file from within the program onto the Master disk.
- Go to the Desktop and drag the file's icon from the Master disk onto Backup disk #1.

From then on, after every use of the program, do the following:

1. Save the file from within the program onto the Master disk.
2. Go to the Desktop, and
   a. If it's Monday, drag the file's icon from Backup disk #1 onto Monday's backup disk.
   b. If it's Tuesday, drag the file's icon from Backup disk #1

onto Tuesday's backup disk.

c. Etc., for the remaining days of the week.

3. Drag the file's icon from the Master disk onto Backup disk #1.

SOME ADDITIONAL PUBLIC DOMAIN AND SHAREWARE PROGRAMS

ARC SHELL

When you buy a public domain disk (or receive a monthly disk from a magazine, or download files from CompuServe, GEnie, or a BBS) you may discover that the files have ".ARC" extensions. "ARC" stands for archive, which means that the file is in compressed form. Before an ".ARC" file can be used, it must be decompressed. Although this involves a slight amount of work on your part, archivied files save you time, space and money. When files are archived, more of them can fit on a disk. And if you want to download or upload files over the phone, archived files (which have fewer bytes than before they were compressed) will take less time to transmit.

The main problem with archived files is that the uninitiated are often confused about how to decompress them. This is not surprising, since the program that is most commonly used for decompression - ARCX.TTP -- requires you to type in, without making a single typing mistake, the name (and sometimes the location) of the file you want to decompress. (See Figure 10.5)

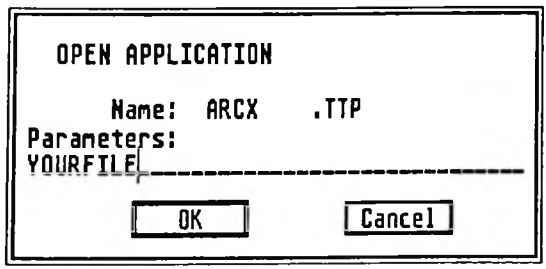


Figure 10.5

But there's an easier way. ARC Shell, by C. F. Johnson, allows you to use the mouse to point and click on the program's various options, rather than having to type in precise information on a command line. (See Figure 10.6) With ARC Shell, you can either extract (i.e., unarc) files that are already compressed, or archive (i.e., compress) regular files. (The program can also copy files [with wild cards and with query], delete and rename files, create and delete folders, show the free space on any connected drive, and run other programs.)

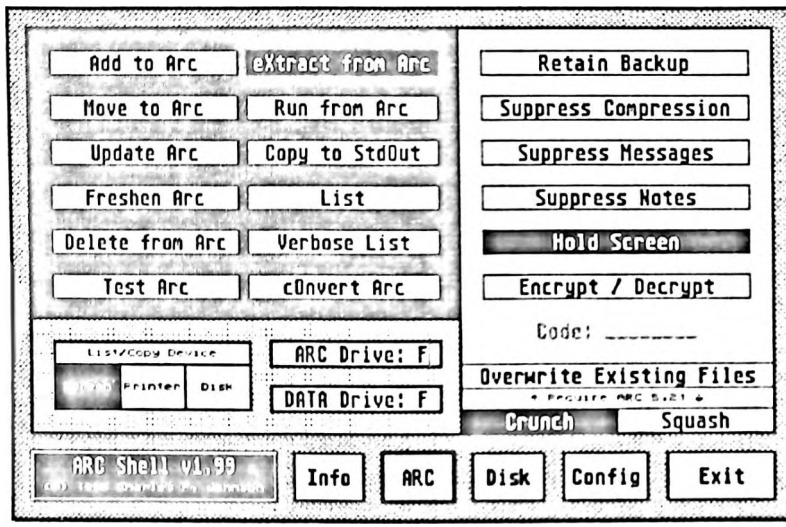


Figure 10.6

In order for ARC Shell to work, you must have the ARC program (ARC . TTP) in the same directory with ARC Shell. In other words, ARC Shell is always used in conjunction with a second program, ARC.TTP. (The actual work of compressing or decompressing is done by ARC.TTP, and ARC Shell just adds a GEM interface to the ARC program.) ARC Shell is SHAREWARE.

QUICKFIND

QUICKFIND, by Ignac A. Kolenko and Derek Mihocka, helps you locate files. (See Figure 10.7) Let's say you want to find a

letter that you wrote to Sue, but you don't know the exact name of the file, nor do you know what disk it's on. With QUICKFIND, you could perform a search of all your drives using the search mask "SUE *.*" This would result in all the file names on your disks that start with "SUE" being listed. If this doesn't locate your file, but you think you may have originally created the file from within a word processor that automatically assigns a "DOC" extension to saved files, you could try a search with "*.DOC" as the mask. With this, QUICKFIND will display all the files that have a "DOC" extension. (* is a "wild card," and is discussed, along with the "?" wild card, in Chapter 4 of the *Atari ST Book*.) QUICKFIND can also delete, write-protect, and update the date stamp of files, as well as create printouts of disk listings. QUICKFIND is SHAREWARE.

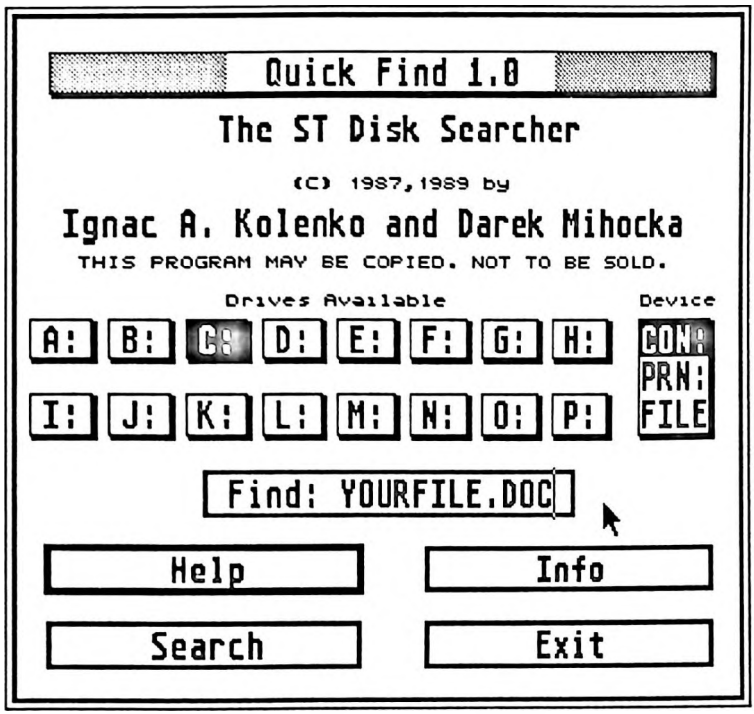


Figure 10.7

ASCII Control Characters on the ST

Dec	Hex	Keyboard	Screen	Name	Description
0	00	^ @		NUL	null character
1	01	^ A		SOH	start of header
2	02	^ B		STX	start of text
3	03	^ C		EXT	end of text
4	04	^ D		EOT	end of transmission
5	05	^ E		ENQ	enquire
6	06	^ F		ACK	acknowledge
7	07	^ G		BEL	bell
8	08	^ H		BS	backspace
9	09	^ I		HT	horizontal tab
10	0A	^ J		LF	line feed
11	0B	^ K		VT	vertical tab
12	0C	^ L		FF	form feed
13	0D	^ M		CR	carriage return
14	0E	^ N		SO	shift out
15	0F	^ O		SI	shift in
16	10	^ P		DEL	delete
17	11	^ Q		DC1	device control 1
18	12	^ R		DC2	device control 2
19	13	^ S		DC3	device control 3
20	14	^ T		DC4	device control 4
21	15	^ U		NAK	negative acknowledge
22	16	^ V		SYN	synchronize
23	17	^ W		ETB	end of text block
24	18	^ X		CAN	cancel
25	19	^ Y		EM	end of medium
26	1A	^ Z		SUB	substitute
27	1B	^ [		ESC	escape
28	1C	^ \		FS	file separator
29	1D	^]		GS	group separator
30	1E	^ 6		RS	record separator
31	1F	^ -		US	unit separator

Dec	Hex	ASCII
32	20	(space)
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	Δ

Dec	Hex		Dec	Hex		Dec	Hex	
128	80	Ç	169	A9	⌐	212	D4	ק
129	81	ü	170	AA	¬	213	D5	ר
130	82	é	171	AB	½	214	D6	ש
131	83	â	172	AC	¼	215	D7	ת
132	84	ä	173	AD	¡	216	D8	ך
133	85	à	174	AE	«	217	D9	ן
134	86	å	175	AF	»	218	DA	ם
135	87	ç	176	B0	ã	219	DB	ף
136	88	ê	177	B1	õ	220	DC	ץ
137	89	ë	178	B2	Ø	221	DD	§
138	8A	è	179	B3	ø	222	DE	∧
139	8B	ï	180	B4	œ	223	DF	∞
140	8C	î	181	B5	Œ	224	E0	α
141	8D	ì	182	B6	À	225	E1	β
142	8E	Ä	183	B7	Ã	226	E2	Γ
143	8F	Å	184	B8	Õ	227	E3	π
144	90	É	185	B9	¨	228	E4	Σ
145	91	æ	186	BA	´	229	E5	σ
146	92	Æ	187	BB	†	230	E6	µ
147	93	ô	188	BC	¶	231	E7	τ
148	94	ö	189	BD	©	232	E8	Φ
149	95	ò	190	BE	®	233	E9	Θ
150	96	û	191	BF	™	234	EA	Ω
151	97	ù	192	C0	ij	235	EB	δ
152	98	ÿ	193	C1	IJ	236	EC	∞
153	99	Ö	194	C2	א	237	ED	φ
154	9A	Ü	195	C3	ב	238	EE	∈
155	9B	ç	196	C4	ג	239	EF	∩
156	9C	£	197	C5	ד	240	F0	≡
157	9D	¥	198	C6	ה	241	F1	±
158	9E	ß	199	C7	ו	242	F2	≥
159	9F	ƒ	200	C8	ז	243	F3	≤
160	A0	á	201	C9	ח	244	F4	⌠
161	A1	í	202	CA	ט	245	F5	⌡
162	A2	ó	203	CB	י	246	F6	÷
163	A3	ú	204	CC	כ	247	F7	≈
164	A4	ñ	205	CD	ל	248	F8	°
165	A5	Ñ	206	CE	מ	249	F9	∙
166	A6	ª	207	CF	נ	250	FA	·
167	A7	º	208	D0	ס	251	FB	√
168	A8	¿	209	D1	ע	252	FC	n
			210	D2	פ	253	FD	2
			211	D3	צ	254	FE	3
						255	FF	

PUBLIC DOMAIN AND SHAREWARE PROGRAMS

The following contains: the program name; page number; CompuServe (CIS) name and location ("AP, L" = "ATARIPRO Forem, Library"); GEnie name and file number; and BRE Software disk number. (CompuServe telephone: 800-848-8199 or 614-457-8600); (GEnie telephone: 800-638-9636 or 301-251-6475) (BRE telephone: 800-347-6760 or 209-432-3072; 352 W. Bedford Ave., Fresno, CA 93711)

ARC Shell, p 147; CIS:(ARSHE199.ARC) AP, L 2; GE:(ARCSH199.ARC) 10973; BRE: 825
ARCTTP, p 147; CIS:(ARCTTP) AP, L 4; GE:(ARCTTP) 1271; BRE: 825
CONTEX.PRG, p 48; BRE: 888
FOLDRXXX.PRG, p 87; CIS:(FOLDER.ARC) AP, L 4; GE:(FOLDRXXX.PRG) 3632; BRE: 802
FORMAT40.TOS, p 128; GE:(FORMAT40.TOS) 180; BRE: 888
HD–WEDGE, p 98; BRE: 888
IBMDISK.DOC, p 124; GE:(IBMDISK.DOC) 550; BRE: 888
L-CACHE.PRG, p 95; CIS:(LUTILS.ARC) AP, L 4; GE:(LUTILS.ARC) 6876; BRE: 395
LITTLE GREEN SELECTOR, p 101; CIS:(LGS14.ARC) AP, L 4; GE:(LGSEL–14.ARC) 12050; BRE: 803
MACCEL2.PRG, p ; CIS:(MACCEL.ARC) AP, L 4; GE:(MACCEL2.ARC) 11615; BRE: 804
MEG-A-MINUTE ELITE, p 100; CIS:(MEGMIN.ARC) AP, L 4; BRE:
MEMFILE, p 37; CIS:AP, L 4; GE:(MEMFIL20.ARC) 7889; BRE: 470
PCF554.PRG, p 125; CIS:(PCF554.PRG) AP, L 4; GE:(PCF554.PRG) 9222; BRE: 888
QUICK FIND, p 149; CIS:(QFIND.ARC) AP, L 4; GE:(QFIND15.ARC) 10686; BRE: 888
RATEHD.PRG, p 89; CIS:(RATEHD.ARC) ATARIVEN, Library 1; GE:(RATEHD.ARC) 12134; BRE: 696
SIXMS.PRG, p 125; CIS:(SIMS.PRG) AP, L 4; GE:(SIMS.PRG) 547; BRE: 888
STWRITER, p 38; CIS:(STW34.ARC) AP, L 5; GE:(STWEL32.ARC) 12495; BRE: 855
ST–IBMDD.PI3, p 124; GE:(ST–IBMDD.PI3) 583; BRE: 888
TERRAPIN, p 101; CIS:(TERRAPIN) AP, L 4; BRE: 888
TINI TOOL, p 22, 65; CIS:(TTOOL.ARC) AP, L 6; GE:(TINYTOOL.ARC) 2236; BRE: 888
TURTLE, p 100; CIS:(TURTLE.ARC) AP, L 4; GE:(TURTLE.ARC) 9397; BRE: 694
UNITERM, p ; CIS:(UNIPRG.ARC) AP, L 2; GE:9684; BRE: 332
WHEREIS.TOS, p 102; CIS:(WHERIS.ARC) AP, L 4; BRE: 888

PRODUCTS AND ADDRESSES

Atari Australia, 376 Lane Cove Rd., North Ryde, NSW 2113, Australia, phone: 02-805-0344

Atari Canada, 90 Gough Rd., Markam, ON, Canada L3R 5V5, phone: 416-479-1266

Atari U.K., Atari House, Railway Terrace, Slough, Berkshire SL2 5BZ, Great Britain

Atari U.S.A., 1196 Borregas Ave., Sunnyvale, CA 94086, phone: 408-745-2367

BRE Software, 352 W. Bedford Ave., Suite 104, Fresno, CA 93711, phone: 800-347-6760 or 209-432-3072

Berkeley Microsystems, P. O. Box 20119, Oakland, CA 94620, phone: 415-547-2191

CompuServe, phone: 800-848-8199 or 614-457-8600

GEnie, phone: 800-638-9636 or 301-251-6475

ICD, Inc., 1220 Rock St., Rockford, IL 61101, phone: 5-968-2228

MichTron, 576 S. Telegraph, Pontiac, MI 48053, phone: 313-334-5700

Supra Corp., 1133 Commercial Way, Albany, OR 97321, phone: 503-967-9075

TEAC, 7733 Telegraph Rd., Montebello, CA 90640, phone: 213-726-0303

Timeworks, 444 Lake Cook Rd., Deerfield, IL 60015, phone: 708-948-9200

VersaSoft Corp., 4340 Almaden Expressway, San Jose, CA 95118, phone: 408- 723-9044

WordPerfect Corp., 1555 N. Technology Way, Orem, UT 84057, phone: 800-321-4566 or 801-222-5670

INDEX

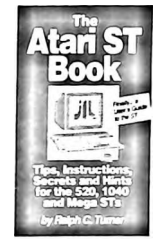
Adaptec controllers, 107-108
"An item with this name..." (alert message), 138
ARC Shell, 147
ARCX.TTP, 147
ARC.TTP, 148
ASCII
  and file editor, 24
  and hexadecimal, 25
  character code, 18
  definition, 75
  extended character set, 37
  foreign characters, 37
  in general, 15
  transfer via modem, 41
ASCII files
  and native format, 43
  definition, 42
  transfer via modem, 42
  weird format, 47
ASCII format
  and file extenders, 46
  save DESKTOP.INF file in, 45
  situations requiring, 44
auto booting, hard drive, 81
AUTO folder, 85
average access time, 88
backup
  failure, 142
  hard drive, 100
  importance of, 135
  procedure, 142
BASIC, 30
bell symbol, 25
bell, printer, 34
binary
  code, 7, 13, 19
  digits, 13
  is hard to read, 19
  numbering system, 20
bits, 13, 18
blank spaces, removing, 49
boot sector, 57, 66, 70
booting, hard drive, 80
byte, 13
cables, 123, 108
caching, 94
carriage return, 25, 47
CD-ROM units, 105
character set, 37
clusters, 56
code, binary, 13
CONTEX.PRG, 48
control characters, 16, 18, 25
  32 non-printing, 18
  appearance, 25
control codes
  altering, 22
  and printer fonts, 38
  embedding, 28, 29
  from specific programs, 38
  lead in and parameter, 35
  multy-byte, 34
  single-byte, 34
  via Control key, 30

controllers
Adaptec, 107
hard drive, 75
jumper plugs, 107
multiple, 105
copyright symbol, 38
corrupted disk, 135
CR symbol, 25
database printing problem, 15
dBMAN, 30
decimal codes, 15, 18, 19
decimal numbering system, 20
desk accessories and hard drives, 84
DeskJet printer fonts, 39
Desktop icons, 81, 82
DESKTOP.INF file, 45, 85
directory
garbage characters in, 134
next to boot sector, 57
where on disk, 66
disk
defective, 135
floppy, 52
garbage characters, 134
IBM format, 122
icons, 78
optimizing, 91
reading and writing, 53
structure, 66
write-protect, 138
driver, hard disk, 80
drives
connecting 5 1/4-inch, 120
logical, 79
multiple hard, 107
physical, 79
drives, 5 1/4-inch
cables, 123
formatting, 127
pull up resistors, 123
step rate, 125
EC symbol, 31, 32
ESC character, 31, 32
Esc key, 36
escape code, 34
escape sequences, 18
exporting files, 44
extended ASCII characters, 18, 38
fan, 118
FAT (see file allocation table)
file
deletion, 140
extenders, 46
fragmentation, 58
read-only status, 138
file allocation table, 57, 60, 66
file editors, 17, 22
changing bytes, 32
MEMFILE, 37
MichTron Utilities, 62
TINI TOOL, 22
file recovery, 51, 62, 65
Five & 1/4-inch drives, 120, 132
folder, can't delete, 137
fonts, 38
formatting
5 1/4-inch drives, 127
creation of sectors, 55
creation of tracks, 54
hard drive, 77
in general, 54
fragmentation, 59
eliminating, 93
file, 58
hard drive, 92
garbage characters, 134
handler, hard drive, 80

hard drive
40-folder limit, 86, 87
advantages of, 73
and desk accessories, 84
and DESKTOP.INF file, 85
auto booting, 81
AUTO folder, 85
average access time, 88
backing up, 100
booting software, 80
caching, 94
capacity, 74
case, 112
characteristics of, 73
components, 75
controllers, multiple, 105
density schemes, 109
fan, 118
formatting, 77, 116
fragmentation, 92
handler, 80
head parking, 99
host adapter mounting bracket, 112
host adapters, multiple, 105
icons, 81
interleave, 89
mechanisms, 118
multiple units, 105
optimizing, 91
partitioning, 78, 95
power supply, 118
reliability, 74
ribbon cable, 118
SCSI bus, 105
Seagate 277N, 111
set up, 77
size, 76
speeding up, 95
suppliers, 119
turning on, 83
useful programs, 101
warnings, 83
head parking, 99
hexadecimal, 15
16 symbols, 20
characters, 25
numbering system, 20
reason for use, 19
required by printer, 18
host adapter, 75, 105, 116
IBM
disk formats, 122
emulation, 129, 132
file format, 47
ICD formatter, 117
icons, disk, 78
importing files, 44
interleave, 89
Item Selector, 101
laser printers, 136
lead in, 35
life feed character, 25
line spacing, 36
Linotype, 136
Little Green Selector, 101
logical drives, 79
logical unit numbers, 108
LUN IDs, 108
margin, printer, 35
MEMFILE, 37
memory, 14
MFM density scheme, 109
MichTron Utilities, 62
modem, 41
mounting bracket, 112
mouse, 141
mouse pad, 142
multi-byte control codes, 34
native format files, 42

Norton rating, 130
null modem, 41
optimizing, hard drive, 91
parameter, control code, 35
partitioning, 78, 95
partitions
  16-meg limitation, 78
  restrictions on, 78
  zeroing out, 79
PC Speed, 130
pc-ditto I, 131
pc-ditto II, 132
physical drives, 79
pin "1" on ribbon cable, 115
power supply, 118
printer
  changing fonts, 38
  compressed printing, 29
  control characters, 16
  double width printing, 29
  line spacing, 36
  right margin, 35
  special features, 29
  stops inserting spaces, 15
printing service bureaus, 136
pull up resistors, 123
random access storage, 54
QUICKFIND, 148
RATEHD.PRG, 89
read-only, 138
reading of disks, 53
recovery, file, 51
Regent Word, 38
resistor terminators, 109
Return key, 27
ribbon cable, 115, 118
RLL density scheme, 109
SCSI
  acronym, 75
  bus, 105
  cables, 107, 115
  devices, 105
  ID numbers, 107, 114
Seagate 277N, 114
sector editor, 51, 65
sector, boot, 57
sectors, 55, 58
sequential storage, 53
SHIFT OUT, 29
single-byte control codes, 34
spaces, removing, 49
ST506/ST412 interface, 108
starting cluster number, 57
step rate software, 125
ST WRITER, 38
SUPERCHARGER, 130
Supra formatter, 117
tape backup, 105
termination resistors, 109
text files, 42
TINI TOOL, 22, 65
tracks, 54
trash can, 59, 140
Universal Item Selector, 101
VIP Professional, 38
word processors, 44
Word Writer, 38, 49
write-protecting disks, 138
writing to disks, 53
zeroing out partitions, 79
zeros (and ones), 12
"$" represents hex value, 33
40-folder limit, 86
^ symbol, 31

Here's What Reviewers and Editors Have Said About *The Atari ST Book*:



STart: "Arguably the best ST users' guide to date. A new user will understand it easily, yet it contains enough tricks and tips to be worthwhile reading even for experienced users. If I owned Atari Corp., I'd make sure to include a copy of this book with every ST sold today." (May 1989 issue. Reviewed by Ron Luks, founder and manager of Atari Forums on CompuServe.)

ST-Log: "A well-written volume full of hints, tips and detailed instructions. Even an experienced hand will be surprised at how much new information is here." (September 1989 issue.)

ST Informer: "The easiest way to describe this book is to call it hard copy of a year's worth of user group meetings run by a master teacher." (July 1988 issue.)

ST World (Published in the U.S.): "This book is like the best of the articles in the ST press, without the hype." (July 1988 issue.)

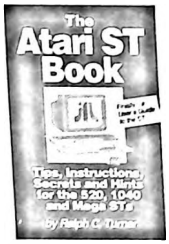
STart: "Besides STart magazine, of course, the single best resource for the novice ST owner is *The Atari ST Book*." (STart editor, October 1989 issue.)

Computer Shopper: " . . . filled to the brim with the information an expert ST user knows, but generally learned on his own." (October 1988 issue.)

Current Notes: " . . . documentation about the ST that is genuinely helpful." (September 1988 issue.)

Antic: " . . . isn't just a rehash of the ST owner's manual . . . Informative, understandable . . . Best of all, it's mostly for the user, not the programmer, although the programmer might learn a thing or two as well." (November 1988 issue.)

The Atari ST Book of Tips, Instructions, Secrets and Hints for the 520, 1040 and Mega STs, By Ralph C. Turner. 158 pages. ISBN 0-945959-00-1. Available from Atari dealers, bookstores, or directly from the publisher (Index Legalis Publishing Company, Post Office Box 1822-3, Fairfield, IA 52556, U.S.A. Phone 515-472-2293)



Also by Ralph C. Turner:

The Atari ST Book

Up until now, the non-programming Atari ST user has been neglected. Although the ST can be a powerful and easy-to-use computer, many of its features and capabilities haven't been adequately explained in language that the average user can understand.

But now, *The Atari ST Book of Tips, Instructions, Secrets and Hints* allows users to take command of their STs. *The Atari ST Book* is aimed at users rather than programmers, and begins where the Atari owners' manuals leave off. The 158-page users' guide provides step-by-step instructions that help both beginning and experienced users get the most out of their STs.

Topics Covered Include

* RAM disks
* Using (or replacing) the Item Selector
* The DESKTOP.INF file and the Control Panel
* Desk Accessories
* Installing an Application
* Printers
* Word Processors
* Desktop Tricks
* The AUTO folder
* Files and their Management
* Copying
* Adjusting the Control Panel's Palette Settings
* Cold Boots and Warm Boots
* Public Domain Software
* Modems and Telecommunications
* and many other subjects

The Atari ST Book. Available from Atari dealers, bookstores, or directly from the publisher: Index Legalis Publishing Company, Post Office Box 1822-3, Fairfield, IA 52556, U.S.A. Phone 515-472-2293

Intermediate and Advanced Atari ST Subjects

At last! A book written in plain English that explains many of the difficult procedures and concepts challenging the serious ST user.

This fascinating new book is packed with step-by-step instructions that help both beginning and advanced users to tap their ST's full potential.

TOPICS COVERED INCLUDE

- Hard disk management and optimization
- ASCII files
- Running IBM programs on your ST
- Disk structure and recovery of lost files
- Assembling your own hard drive system from inexpensive components
- SCSI ID numbers and LUN IDs
- Connecting multiple SCSI devices
- Connecting a 5¼-inch floppy disk drive
- Sector and file editing
- Binary, hexadecimal and decimal codes
- Control codes and escape sequences
- And many miscellaneous subjects

Index Legalis Publishing Company
Post Office Box 1822-3
Fairfield, IA 52556 U.S.A.
(515) 472-2293

Atari, ST and the Atari logo are registered trademarks of the Atari Corporation

ISBN 0-945959-01-X